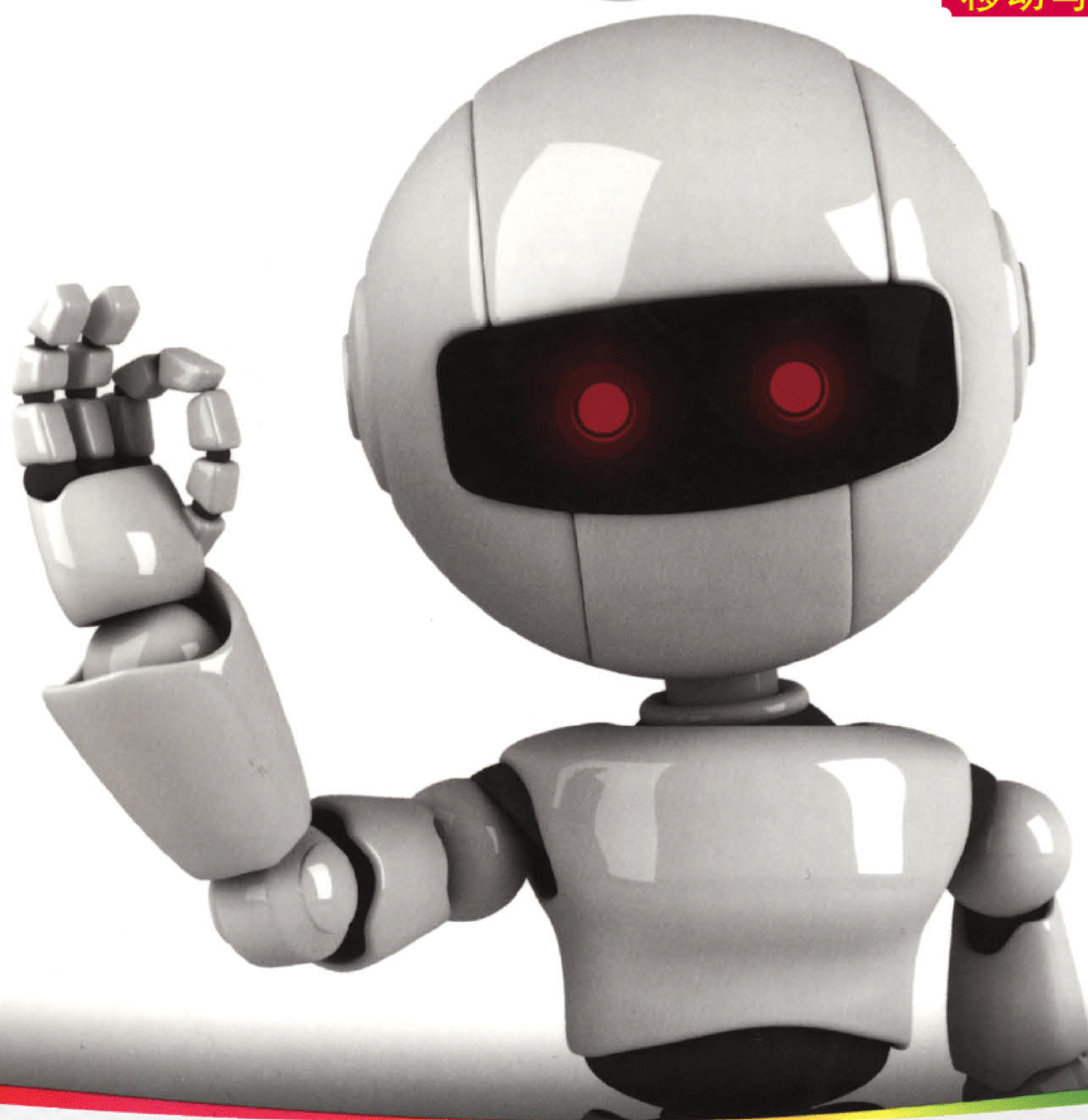
Join the discussion @ p2p.wrox.com



Wrox Programmer to Programmer™

移动与嵌入式开发技术



Beginning Android Application Development

Android 编程入门经典

[美] Wei-Meng Lee 著
何晨光 李洪刚 译

清华大学出版社

移动与嵌入式开发技术

Android编程入门经典

[美] Wei-Meng Lee 著
何晨光 李洪刚 译

清华大学出版社
北 京

Wei-Meng Lee
Beginning Android Application Development
EISBN：978-1-118-01711-1

北京市版权局著作权合同登记号 图字：01-2011-4831

图书在版编目(CIP)数据

Android 编程入门经典 / (美) 李伟梦(Wei-Meng Lee) 著；何晨光，李洪刚 译.
—北京：清华大学出版社，2012.4

书名原文：Beginning Android Application Development

(移动与嵌入式开发技术)

ISBN 978-7-302-28340-9

I. ①A… Ⅱ. ①李… ②何… ③李… Ⅲ. ①移动终端—应用程序—程序设计 Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字(2012)第 047748 号

责任编辑：王 军
装帧设计：牛艳敏
责任校对：邱晓玉
责任印制：

出版发行：清华大学出版社
网 址：http://www.tup.com.cn，http://www.wqbook.com
地 址：北京清华大学学研大厦 A 座 **邮 编：**100084
社 总 机：010-62770175 **邮 购：**010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经 销：全国新华书店
开 本：185mm×260mm **印 张：**24.25 **字 数：**636 千字
版 次：2012 年 4 月第 1 版 **印 次：**2012 年 4 月第 1 次印刷
印 数：1～4000
定 价：49.00 元

产品编号：

译　者　序

正如我们所看到的，以手机为代表的移动通信设备的发展日新月异，给人们的生产生活方式带来了革命性的影响。从上世纪90年代初只具有单一功能且使用模拟信号的“大哥大”，到如今代表了智能手机终端综合设计和应用水平的业界翘楚iPhone 4，都无不彰显了技术和市场的力量。而这一变革的原动力归根结底就是对自由和开放的不懈追求——技术的自由，市场的开放，使更多的人能够加入到这一潜力巨大的智能手机业务的开发领域。由此催生的多姿多彩的手机操作系统平台和五花八门的各类应用软件，带给了我们更多的繁荣和丰富的用户体验。而Android手机操作系统正是这一开放平台的代表，它短短5、6年的发展历史已经使得苹果公司大为头痛，不得不派乔布斯去上帝那儿寻求帮助了，更遑论诺基亚之类抱残守缺的徒劳挣扎；曾经的业界大佬们如三星、摩托罗拉也终于发现了对抗苹果的利器而对其趋之若鹜。

的确，现在的Android借助于Google“开放手机联盟”的巨大影响，由于其开放性，已经充斥了日常生活的方方面面。随着Android设备的进一步普及，因其可扩展性和便于携带的特征，Android智能终端正迅速地融入生活，甚至可以与微博、Facebook、Twitter进行无缝连接。尤其在移动健康监护领域，已经逐渐催生出以此为目的的特种手机配件和应用服务等一系列新兴产业。若能结合云计算如火如荼的发展之势，相信这将又是一个创造新的产业增长点的大好时机。译者近年来也有幸参与了与之有关的系列研发工作，深知其中甘苦，因此很乐意将本书介绍给广大读者，来共同迎接新的IT产业革命的到来。

本书是一本有关基于Android手机操作系统进行应用程序开发的入门读物，作者Wei-Meng Lee在移动操作系统平台的项目开发和培训上具有丰富的实践经验。他采用图文并茂、上手性极强的步步引导的方式将一个门外汉领入Android的大千世界的同时，又为其展示了较为广阔的视野，避免了初学者常常具有的只见树木不见森林的缺憾，堪称一大特色。本书从Android的发展沿革讲起，通过对其中关键概念深入浅出的介绍，用大量的示例说明了Android应用程序的构成、表现形式以及运行原理，为读者描绘了较为完整的Android开发蓝图。在此基础上，引入了一些高级组件和功能的介绍，为读者进一步的实践和开发高价值的应用程序指明了方向。除辅以每章课后的练习使读者巩固所学之外，通篇口语化的表达方式也拉近了本书与读者的距离。当然，由于是面向初学者，Android本身的体系结构和原理并未过多介绍，对于想对此有更深入了解的读者未免有意犹未尽之感；同时，也是广大读者比较关心的，对于如何通过编写应用程序来获取经济效益，本书只是简单提到了“经典”方法，没有涉及更为灵活的技术手段，略有遗憾。当然，这也是可以理解的，毕竟本书不是一本专注于市场营销的书籍。

在全书翻译过程中，对于Android中最重要的两个概念——activity和intent，考虑到译本是面向国内读者的入门读物，为了更好地建立自己的汉语语境和术语体系并利于读者理解，我们统一将其分别翻译成“活动”和“意图”。尤其是在原理性说明或描述性段落中，但是在与代码相关的上下文中仍旧保持原先的英文形式，读者阅读时请注意这一点。其他的相关专有名词也作类似处理，包括章节标题。

限于译者水平，译文定有很多不当之处，敬请读者批评指正。

何晨光　李洪刚

于深圳西丽大学城

作 者 简 介

Wei-Meng Lee是Developer Learning Solutions公司(www.learn2develop.net)的创始人和技术专家，这家技术公司专门从事最新移动技术的培训。Wei-Meng具有多年的培训经验，他的培训课程特别强调实践学习法。这种动手学习编程的方法比通过阅读书籍、教程和文档来理解主题要容易得多。

Wei-Meng还是*Beginning iOS 4 Application Development*(Wrox)等书的作者。读者可以通过weimenglee@learn2develop.net与他联系。

技术编辑简介

Kunal Mittal是Sony Pictures Entertainment公司的技术执行总监，他负责SOA、Identity Management以及Content Management项目。Kunal还是一位帮助创业者确定技术战略、产品路线和开发计划的企业家。他一般担任顾问或咨询公司首席技术官的职位，并从事项目经理和技术架构师的工作。

他已经撰写和编辑了多本关于J2EE、云计算和移动技术方面的专著以及多篇文章。他拥有软件工程的硕士学位，还是一名仪表飞行等级的私人飞行员。

致　　谢

每次完成一本书的编写工作，我总是告诉自己这是我写的最后一部书了。因为写书真的是一个特别费时费力的工作。然而，当你收到读者的电子邮件告诉你他们从中学到了新的技术并为此表示感谢时，所有的挫折都烟消云散了。

果然，当完成前一本关于iOS编程的书后，我又立即签署了另外一本书的编写工作——这次是关于Android的。尽管您在封面上只能看到作者的名字，但实际上是更多的幕后工作者才使本书的出版成为可能。现在本书已经完成，我要对他们表示衷心的感谢。

首先，特别感谢本书的编辑Ami Sullivan，同她合作一直都很愉快。难以置信的是我们在一个非常短的时期内(仅仅一年)已经合作过3本书，本书是第4本了！当我得知Ami将成为我的编辑时，我就知道这个项目将顺利完成。谢谢Ami的悉心指导，谢谢Ami在本书看起来永远不能按期完成的那段时间所具有的耐心。

还不能忘记的是那些幕后的英雄：文字编辑Luann Rouff和技术编辑Kunal Mittal。他们在编辑本书时所具有的锐利眼光，保证了每一句话的准确性——无论是在语法方面还是技术方面。Luann和Kunal，谢谢你们!

我还要借此机会向我在MobiForge.com的编辑Ruadhan O'Donoghue表示感谢，他一直对我的写作非常支持。他总是接受我的想法并且在我的进度落后时给予理解。Ruadhan，谢谢您维护了这么棒的一个网站。

最后，但并非最不重要的一点是，我要感谢我的父母以及妻子Sze Wa所给予我的全力支持。在我写作本书的那段时间，他们无私地调整自己的日程安排来迁就我。Sze Wa在我为满足期限要求拼命工作的无数个夜晚总是一直陪我熬夜，为此我非常感激。另外，我们可爱的小狗Ookii，谢谢你陪伴着我们(如果想知道Ookii是谁，读者可以在本书中找到它的两张照片。找到它们就作为我留给你们的额外练习吧)。

前　　言

我最开始玩Android SDK是在其正式版本1.0发布以前。那时，工具还不完善，SDK中的API不稳定，文档也很缺乏。经过两年半时间的快速发展，现在的Android已经成为一个和iPhone相比毫不逊色的强大的移动操作系统。由于经历过Android成长的所有痛苦，我想现在是开始学习Android编程的最好时机——API已经稳定，工具也有了改善。但是仍存在一个挑战：对许多人来说，入门仍是一个可望而不可及的目标。这一挑战在我脑海里徘徊许久，也成为了我写本书的动力，它也许可以给Android初级程序员带来益处，并使他们能够逐步编写更复杂的应用程序。

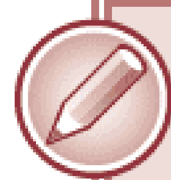
由于本书是写给Android初级开发人员的，为的是使他们能够快速上手，因此我以线性方式涵盖了必要的主题，这样可以使您建立起自己的知识体系而不会被细节淹没。我采取的哲学观点是：最好的学习方法是实践——因此，每一章的“试一试”部分将首先教您如何构建一些东西，然后解释其工作原理。

尽管Android编程是一个宏大的主题，但本书要实现三重目标：帮助读者从最基本的原理入手，使读者理解SDK的底层架构以及领会事情要按特定方式完成的原因。本书超越了目前任何一本面面俱到的有关Android编程的书籍的范围，但我确信当您阅读完此书(并做了练习)之后，将有充分的准备来应对下一个Android编程的挑战。

本书读者对象

本书针对的是打算使用Google的Android SDK来开发应用程序的Android初级开发人员。为了从本书中真正获益，您应该在编程方面具有一些背景知识，并且至少熟悉面向对象编程的概念。如果对Java(Android开发所用的语言)一无所知，那么您也许应该首先学习一门Java编程课程，或者阅读有关Java编程方面的优秀书籍。以我的经验，如果您已经了解C#或VB.NET，学习Java就比较轻松；只要按照“试一试”的步骤就可以使您的学习过程顺利进行。

对于那些对所有编程概念都一无所知的人来说，我知道开发移动应用程序并赚到钱是很有诱惑力的。然而，在尝试本书的示例之前，我想首先学习一些基本的编程知识才是更好的着手点。



注意：本书中讨论的所有示例均使用Android SDK 2.3版本编写和测试。尽管我们已经努力保证本书中所有用到的工具都是最新的，但当您阅读本书时，还是很可能有更新版本的工具可用。如果是这样，某些指示和/或屏幕截图会有少许不同。不过，任何改变都应是可控的。

本书主要内容

本书涵盖了使用Android SDK进行Android编程的基本概念，共分为11章和3个附录。

“第1章：Android编程入门”介绍了Android操作系统的基本概念和当前发展状况。您可以了解到Android设备的各种功能以及市场上一些比较流行的设备。还可以学习如何下载和安装所有必需的工具来开发Android应用程序并在Android模拟器上进行测试。

“第2章：活动和意图”使您熟悉Android编程中的两个最重要的概念：活动和意图。活动是Android应用程序的构建块。您将学习如何使用意图将活动链接起来形成一个完整的Android应用程序。意图是链接活动的胶水，也是Android操作系统的独特特征之一。

“第3章：Android用户界面”介绍了Android应用程序的用户界面的不同组成部分。您将学习到用来构建应用程序的用户界面的不同布局，以及当用户和应用程序交互时与用户界面相关联的多种事件。

“第4章：使用视图设计用户界面”介绍了可用于构建Android用户界面的各种基本视图。该章将学习3组主要的视图：基本视图、选取器视图和列表视图。

“第5章：使用视图显示图片和菜单”继续研究视图。您将了解到如何使用不同的图像视图来显示图像，以及在应用程序中显示选项和上下文菜单。该章最后将额外介绍一些很酷的视图，可以用它们来为您的应用程序锦上添花。

“第6章：数据持久化”教您如何在Android应用程序中保存或存储数据。除了学习使用不同的技术来存储用户数据外，您将学习到文件操作以及如何把文件保存到内部或外部存储器(SD卡)上。此外，还将学习到如何在Android应用程序中创建和使用SQLite数据库。

“第7章：内容提供者”讨论了在Android设备的不同应用程序间如何共享数据。您将学习如何使用内容提供者并自己创建一个。

“第8章：消息传递和联网”研究了移动编程中最有趣的两个主题——发送SMS消息和网络编程。您将学习如何以编程方式发送和接收SMS消息，如何连接到Web服务器来下载数据。最后，还将了解在Android应用程序中是如何访问Web服务的。

“第9章：基于位置的服务”描述了如何使用Google Maps来构建基于位置的服务应用程序。您还将学习到如何获取地理位置数据并在地图上显示该位置。

“第10章：开发Android服务”将向您展示如何使用服务来编写应用程序。服务是运行于后台且没有用户界面的应用程序。您将了解如何在一个单独的线程中以异步方式运行您的服务，以及活动与之通信的方法。

“第11章：发布Android应用程序”讨论了您在准备好发布Android应用程序时可以采用的不同方法。您还将了解到在Android Market上发布并出售应用程序的步骤。

“附录A：使用Eclipse进行Android开发”简要概述了Eclipse中的许多功能。

“附录B：使用Android模拟器”提供了有关使用Android模拟器进行应用程序测试方面的一些提示和技巧。

“附录C：练习答案”包含了每章最后的练习的答案。

本书的结构

本书将学习Android编程的任务分解为若干个更小的环节，使您能够在钻研更高级的内容之

前消化每一个主题。

如果您对于Android编程完全是个新手，那就首先从第1章开始。一旦熟悉了基本概念，就可以转到附录去阅读更多有关Eclipse和Android模拟器的知识。当完成这些之后，可以再从第2章继续，并按部就班地学习更高级的主题。

本书一大特色就是每章的所有示例代码都独立于先前章节所讨论的内容。这样，您可以灵活地转入到所感兴趣的主题并按照"试一试"的项目内容开始练习。

使用本书的前提条件

本书中的所有示例都在Android模拟器(作为Android SDK的一部分)上运行。当然，为了从本书中得到更多收获，拥有一个真实的Android设备还是很有益的(尽管这不是绝对必要的)。

源代码

由于需要从头至尾运行本书中的示例，您可以选择手工键入全部代码或者使用本书配套的源代码文件。本书用到的所有源代码文件在www.wrox.com上均可下载。在这个网站上，直接找到本书的书名(使用搜索框或从书名列表选择)，然后在本书的详细页面上点击Download Code链接来获得本书所有的源代码。

所需项目文件的名称将出现在一个代码注释中，它位于"试一试"内容的开头部分，像这样：

代码片段的文件名称

代码下载完成后，使用您熟悉的压缩工具解压缩。或者转到Wrox的代码下载主页www.wrox.com/dynamic/download.aspx上查看本书以及其他所有Wrox书籍的可用代码。

读者也可从本书的支持网站http://www.tupwk.com.cn/downpage上下载本书源代码。对于本书如有任何意见和建议，请发送邮件至wkservice@vip.163.com。

目　　录

第1章

Android编程入门

本章将介绍以下内容

- Android简介
- Android版本及其功能集
- Android架构
- 市场上的各种Android设备
- Android Market应用程序商店
- 如何获得开发Android应用程序的工具和SDK(软件开发工具包)
- 如何开发您的第一个Android应用程序

欢迎阅读本书！既然您手中拿着这本书(或正在您的最新移动设备上阅读它)，那就说明您对学习如何为Android平台编写应用程序很感兴趣——现在就是学习Android应用程序开发的最佳时机。移动应用市场正在迅速增长，最近的市场调查显示，Android已经超越iPhone在美国智能手机市场占据第二的位置。目前第一名的荣誉属于Research In Motion(RIM)，Apple的iPhone是第三名。就在您阅读本书的时候，Android很有可能已经成为美国第一大智能手机平台，而且您也许正在最新的一款Android设备上阅读本书。

是什么使得Google在2005年买入的这个相对不那么知名的操作系统在今天却如此受欢迎呢？它又提供了哪些功能？本章将介绍Android到底是什么，以及是什么让开发人员和设备制造商都有如此大的兴趣。您也将开始开发您的第一个Android应用程序，并学会如何获得必要的工具并对其设置。在本章结尾，您将具备进一步探索更尖端的技术和技巧以开发您的下一个杀手级的Android应用程序所需的基础知识。

1.1 Android简介

Android是一款基于Linux修订版本的移动操作系统。它最初是由同名的Android有限公司作为进入移动市场的战略的一部分于2005年开发的。Google收购了Android公司，并接管了它的开发工作(包括整个开发团队)。

Google要求Android系统是开放和免费的。因此，大部分Android代码在Apache License开源协议下都公开了，这意味着任何想使用Android的人都可以下载Android的全部源代码。此外，供应商(特别是硬件制造商)可以添加他们自己专有的Android扩展，通过定制Android以区别于其他

厂商的产品。这一简单的开发模型使Android非常有吸引力，并因此引起了许多供应商的兴趣。Apple公司iPhone产品的巨大成功彻底改变了智能手机产业，这深深影响到了诸如摩托罗拉和索爱这一类多年只开发自己的移动操作系统的公司。当iPhone发布时，这些大部分的厂商不得不争相寻找振兴自己产品的新出路。他们将Android视为一种解决方案——继续设计自己的硬件，同时将Android用作操作系统并增强其功能。

使用Android的主要优势是它提供了统一的应用程序开发方法。开发人员只需要为Android进行开发，开发出的应用程序可以运行在许多不同的设备上，只要这些设备用的是Android系统。在智能手机界，应用程序是成功链中的最重要一环。因此，为了应对已经占据大量应用程序市场的iPhone带来的巨大冲击，设备制造商对Android寄予了厚望。

1.1.1 Android版本

自首次发布以来，Android已历经了相当多数量的更新版本。表1-1列出了Android的不同版本及其相应代号。

表1-1 Android版本简史

Android版本	发布日期	代号
1.1	2009年2月9日	
1.5	2009年4月30日	Cupcake(纸杯蛋糕)
1.6	2009年9月15日	Donut(炸面圈)
2.0/2.1	2009年10月26日	Eclair(长松饼)
2.2	2010年5月20日	Froyo(冻酸奶)
2.3	2010年12月6日	Gingerbread(姜饼)
3.0	尚未确定(截止到写作本书时)	Honeycomb(蜂巢)

1.1.2 Android功能

鉴于Android的开源以及制造商可对其自由定制的特点，因此没有固定的软硬件配置。然而，Android本身支持如下功能：

- 存储——使用SQLite(轻量级的关系数据库)进行数据存储，第6章将对数据存储进行详细讨论。
- 连接性——支持GSM/EDGE、IDEN、CDMA、EV-DO、UMTS、Bluetooth(包括A2DP和AVRCP)、WiFi、LTE和WiMAX。第8章将详细讨论联网。
- 消息传递——支持SMS和MMS，也在第8章进行详细探讨。
- Web浏览器——基于开源的WebKit，并集成Chrome的V8 JavaScript引擎。
- 媒体支持——支持以下媒体：H.263、H.264(在3GP或MP4容器中)、MPEG-4 SP、AMR、AMR-WB(在3GP容器中)、AAC、HE-AAC(在MP4或3GP容器中)、MP3、MIDI、Ogg Vorbis、WAV、JPEG、PNG、GIF和BMP。
- 硬件支持——加速度传感器、摄像头、数字式罗盘、接近传感器和全球定位系统(GPS)。
- 多点触摸——支持多点触摸屏幕。
- 多任务——支持多任务应用。
- Flash支持——Android 2.3支持Flash 10.1。
- tethering——支持作为有线/无线热点实现Internet连接共享。

1.1.3 Android架构

为了理解Android的工作方式，可以参看图1-1，该图描述了构成Android操作系统(OS)的各个层。

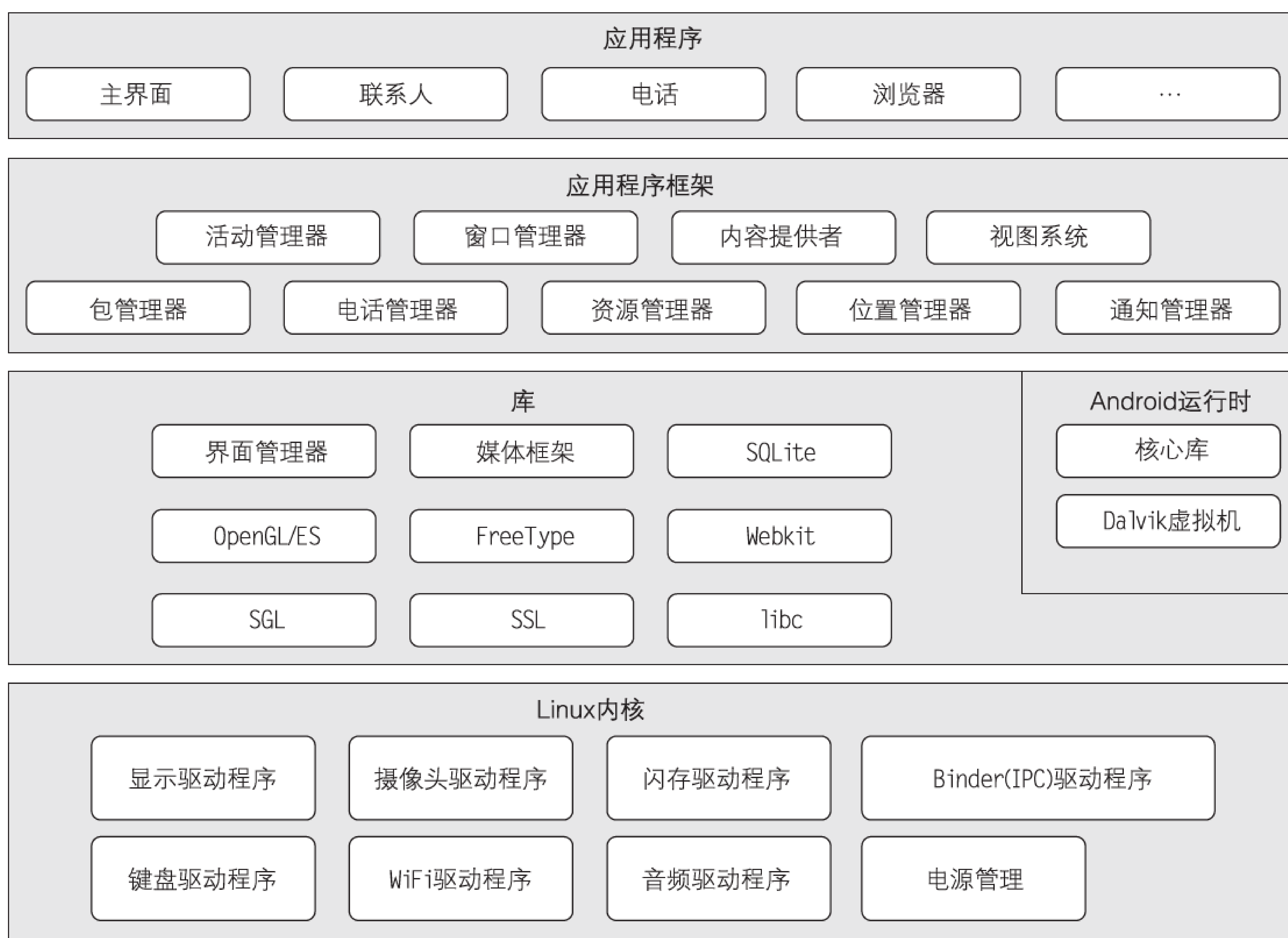


图 1-1

Android操作系统大致可以在4个主要层面上分为以下5个部分：

- Linux内核——这是Android所基于的核心。这一层包括了一个Android设备的各种硬件组件的所有低层设备驱动程序。
- 库——包括了提供Android操作系统的主要功能的全部代码。例如，SQLite库提供了支持应用程序进行数据存储的数据库。Webkit库为浏览Web提供了众多功能。
- Android运行时——它与库同处一层，提供了一组核心库，可以使开发人员使用Java编程语言来写Android应用程序。Android运行时还包括Dalvik虚拟机，这使得每个Android应用程序都在它自己的进程中运行，都拥有一个自己的Dalvik虚拟机实例(Android应用程序被编译成Dalvik可执行文件)。Dalvik是特别为Android设计，并为内存和CPU受限的电池供电的移动设备进行过优化的专门的虚拟机。
- 应用程序框架——对应用程序开发人员公开了Android操作系统的各种功能，使他们可以在应用程序中使用这些功能。
- 应用程序——在这个最顶层中，可以找到Android设备自带的应用程序(例如电话、联系人、浏览器等)，以及可以从Android Market应用程序商店下载和安装的应用程序。您所写的任何应用程序都处于这一层。

1.1.4 市场上的Android设备

Android设备有各种样式和大小。截至2010年11月底，Android操作系统可以支持如下类型的设备：

- 智能手机
- 平板电脑
- 电子阅读器
- 上网本
- MP4播放器
- 互联网电视



图 1-2

而您目前很可能已经至少拥有其中一种设备。图1-2(顺时针)展示了Samsung Galaxy S、HTC Desire HD以及LG Optimus One智能手机。

制造商都趋之若鹜的另一类流行的设备是平板电脑。平板电脑的尺寸通常是7英寸大小(对角线长度)。图1-3展示了Samsung Galaxy Tab和Dell Streak(5英寸的平板手机)。

除了智能手机和平板电脑外，Android也开始出现在专用设备中，如电子书阅读器。图1-4展示了一款运行Android操作系统的彩色电子书阅读器产品——Barnes & Noble公司的NOOKcolor。



图 1-3



图 1-4

除了这些流行的移动设备，Android也正慢慢进入到您的客厅。瑞典公司People of Lava开发了一款基于Android的电视机，名为Scandinavia，如图1-5所示。

Google还涉足了基于Android的专有的智能电视平台，并和诸如英特尔、索尼、罗技等公司进行共同开发。图1-6展示了索尼公司的Google电视。



图 1-5



图 1-6

1.1.5　Android Market

如前所述，决定一个智能手机平台成功的主要因素之一是支持它的应用程序。从iPhone的成功可以清楚地看出，应用程序在决定一个新的平台是成功还是失败方面扮演了一个非常关键的角色。此外，使这些应用程序能为广大用户访问也是极为重要的。

因此，在2008年8月，Google宣布将在同年10月份为用户提供一个适用于Android设备的在线应用程序商店：Android Market。使用预装于Android设备上的Market应用程序，用户可以很方便地把第三方应用程序直接下载到他们的设备上。付费和免费的应用程序在Android Market上都是受支持的，不过付费的应用程序由于法律问题只提供给某些国家的用户。

同样，在一些国家，用户可以从Android Market购买付费的应用程序，但开发人员不能在该国销售。例如，在写作本书时，印度的用户可以从Android Market购买应用程序，但印度的开发人员却不能在Android Market上出售应用程序。相反的情况也可能是存在的。例如，韩国的用户不能购买应用程序，但韩国的开发人员可以在Android Market上出售应用程序。

第11章讨论了更多有关Android Market的内容，以及如何在上面出售自己的应用程序。

1.2　获得所需工具

既然已了解了Android的概念和其功能集，您也许渴望亲自动手试一试，并开始写些应用程序。然而，在您写第一个应用程序之前，需要下载所需的工具和SDK。

对于Android开发，可以使用Mac、Windows PC 或Linux机器。所有必需的工具都可以通过网络免费下载。除了少数需要访问硬件的例子以外，本书提供的大多数例子都可以在Android模拟器中运行得很好。本书中，我将使用运行Windows 7操作系统的计算机来演示所有的代码示例。如果您用的是Mac或Linux计算机，除了存在一些细微的差别，屏幕截图应该是很相似的，您应该可以毫无困难地按照本书的指导来练习。

那么，让我们开始有趣的学习之旅吧！

Java JDK

Android SDK使用Java SE开发工具包(JDK)。因此，如果您的计算机上没有安装JDK，那么应该通过www.oracle.com/technetwork/java/javase/downloads/index.html地址下载并在进入下一小节前进行安装。

1.2.1　Eclipse

开发任何应用程序的第一步都是获得集成开发环境(IDE)。就Android来说，推荐使用Eclipse。它是一个多语言的软件开发环境，有一个可扩展的插件系统。通过它可以用Java、Ada、C、C++、COBOL、Python等语言开发各种类型的应用程序。

对于Android的开发，要下载Eclipse IDE for Java EE Developers (www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/heliossr1)。目前有6个版本可用：Windows(32位和64位)、Mac OS X (Cocoa 32和64) 以及Linux (32位和64位)。只要选择与您的操作系统相对应的那个版本进行安装即可。本书所有示例均使用Windows平台下的32位版本的Eclipse进行过测试。

下载Eclipse IDE后，把其内容(eclipse文件夹)解压到一个文件夹下，比如C:\Android\。图1-7显示了eclipse文件夹的内容。

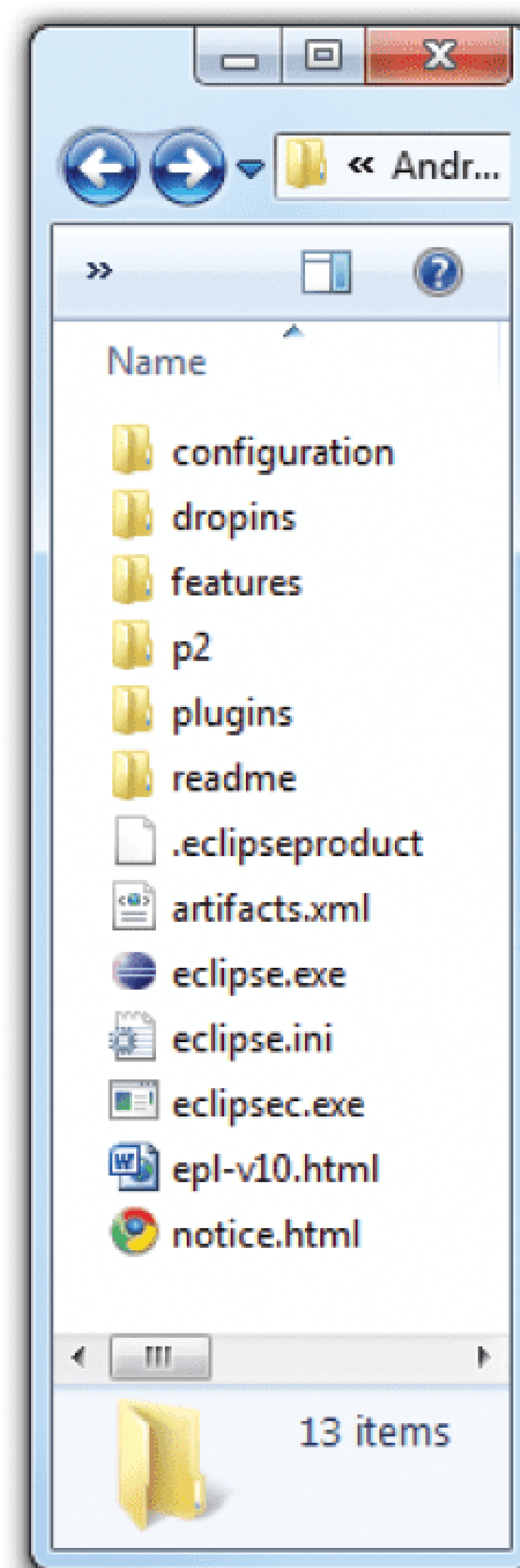


图 1-7

1.2.2　Android SDK

接下来需要下载的一个重要软件自然是Android SDK。Android SDK包含了一个调试器、库、一个模拟器、文档、示例代码和教程。

可以从http://developer.android.com/sdk/index.html下载Android SDK。

一旦SDK下载完成，将其内容(android-sdk-windows文件夹)解压到C:\Android\文件夹下，或者任意一个已经创建好的文件夹下。

1.2.3　Android开发工具

用于Eclipse的Android开发工具(Android Development Tools，ADT)插件是对Eclipse IDE的扩展，用以支持Android应用程序的创建和调试。使用ADT，可以在Eclipse中做如下工作：

- 创建新的Android应用程序项目
- 访问Android模拟器和设备的存取工具
- 编译和调试Android应用程序
- 将Android应用程序导出到Android包(APK)
- 创建数字证书来对APK进行代码签名

为了安装ADT，首先须双击eclipse文件夹下的eclipse.exe文件，启动Eclipse。

当Eclipse首次启动时，系统将提示一个文件夹用作您的工作区。在Eclipse中，工作区是一个用于存储所有项目的文件夹。采用默认的建议，并单击OK按钮。

一旦Eclipse启动和运行后，选择Help | Install New Software...菜单项(如图1-8所示)。

在出现的Install窗口中的文本框内输入http://dl-ssl.google.com/android/eclipse(如图1-9所示)，接着单击Add...按钮。

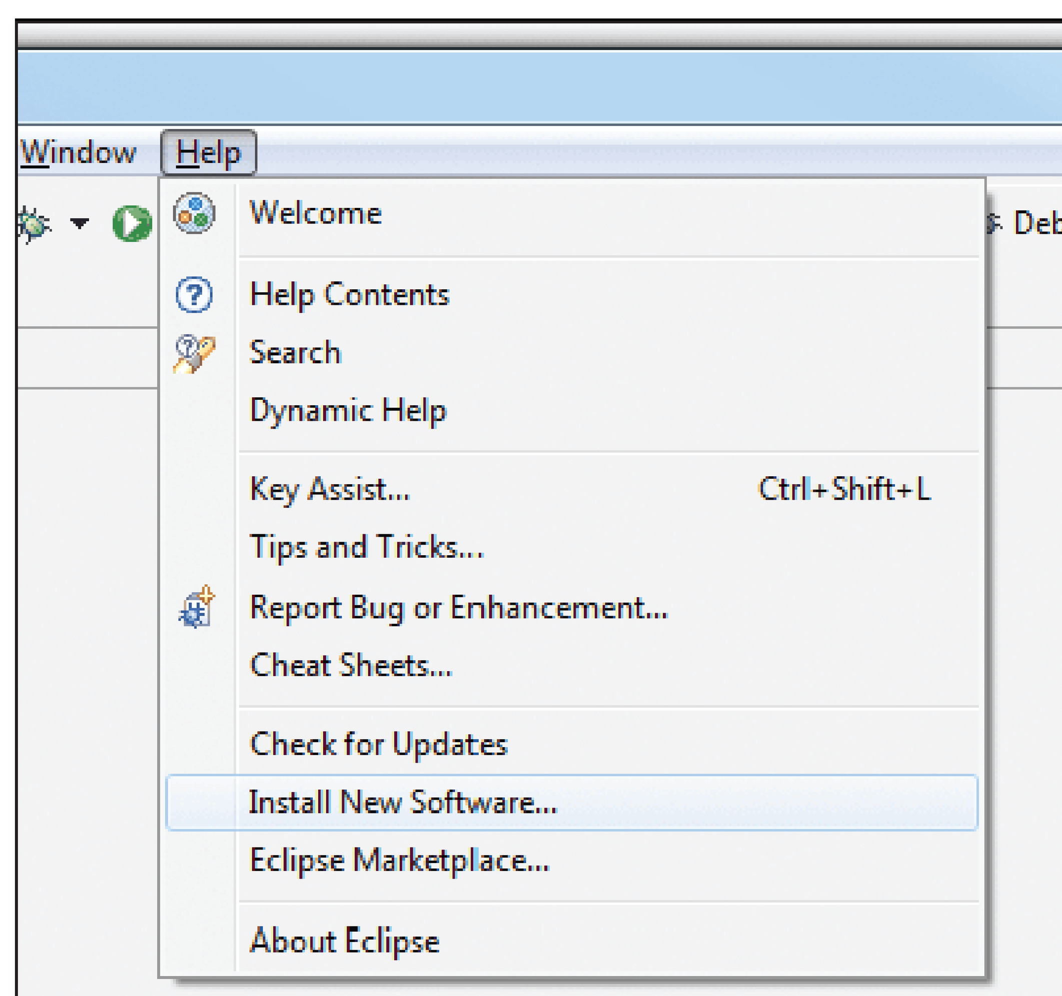


图 1-8

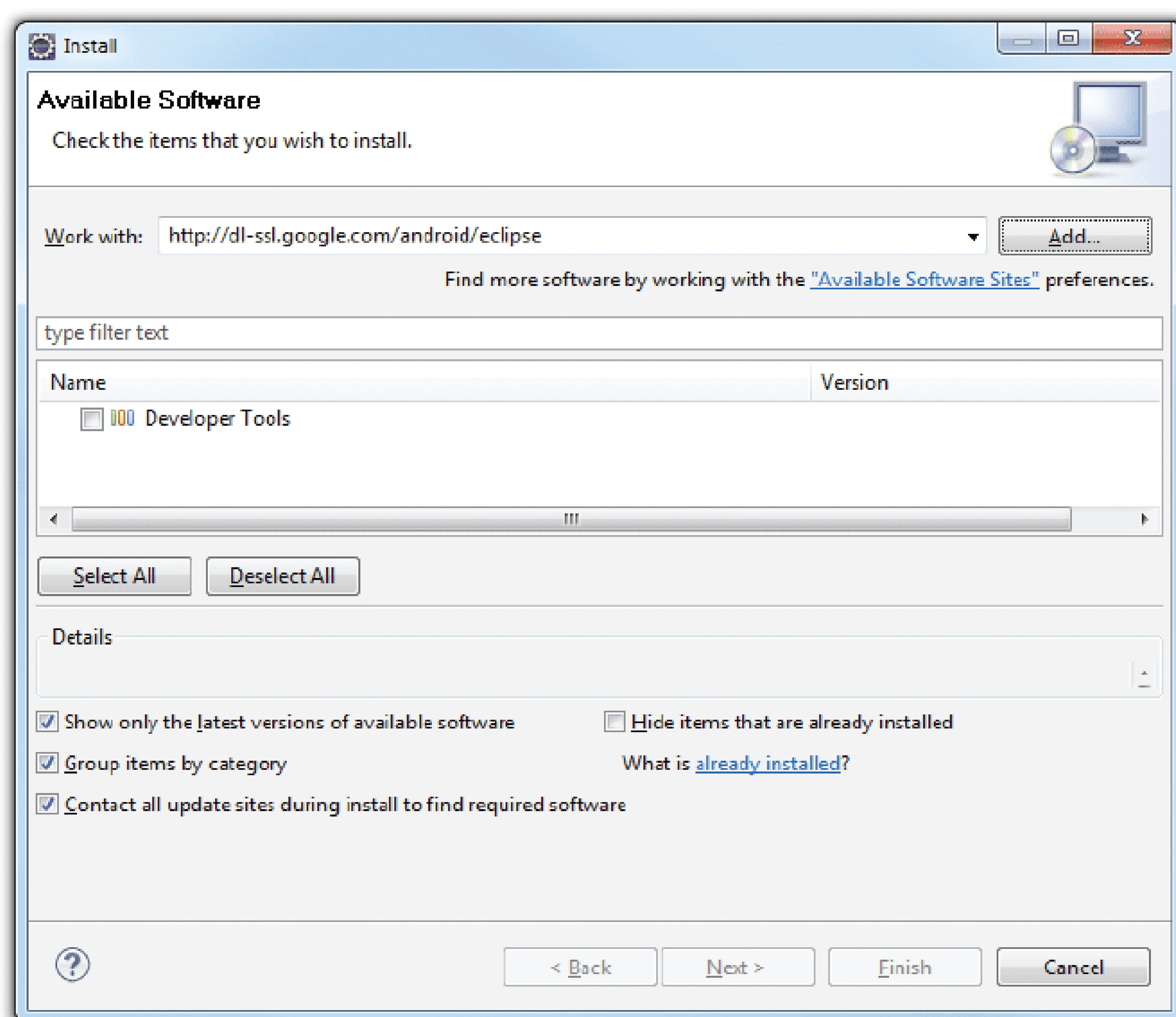


图 1-9

稍候，您将看到在窗口的正中央显示出Developer Tools项(如图1-10所示)。展开后，显示出以下内容：Android DDMS、Android Development Tools和Android Hierarchy Viewer。全选并单击Next按钮。

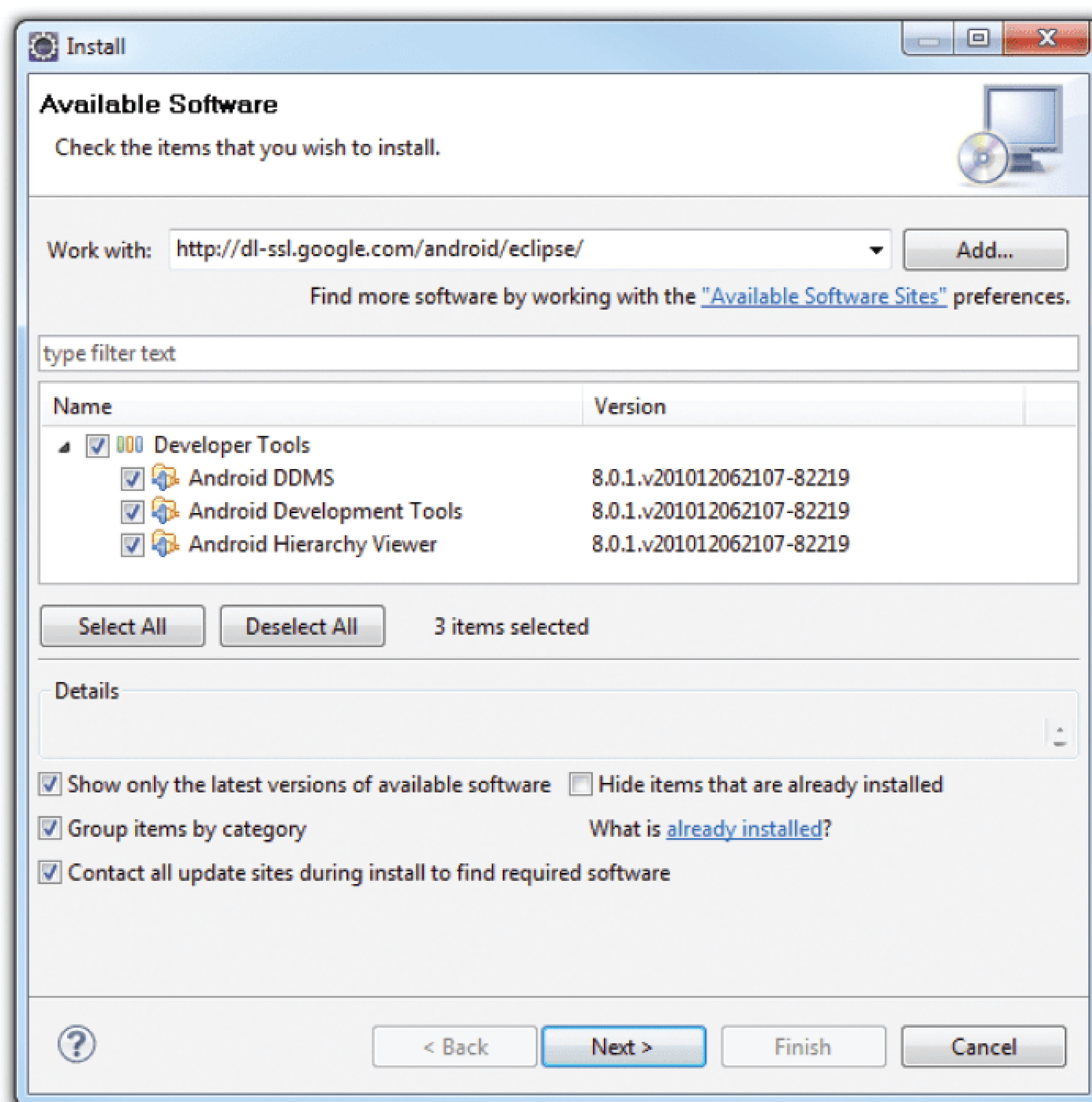


图 1-10

在看到安装详细信息(如图1-11所示)后，单击Next按钮。

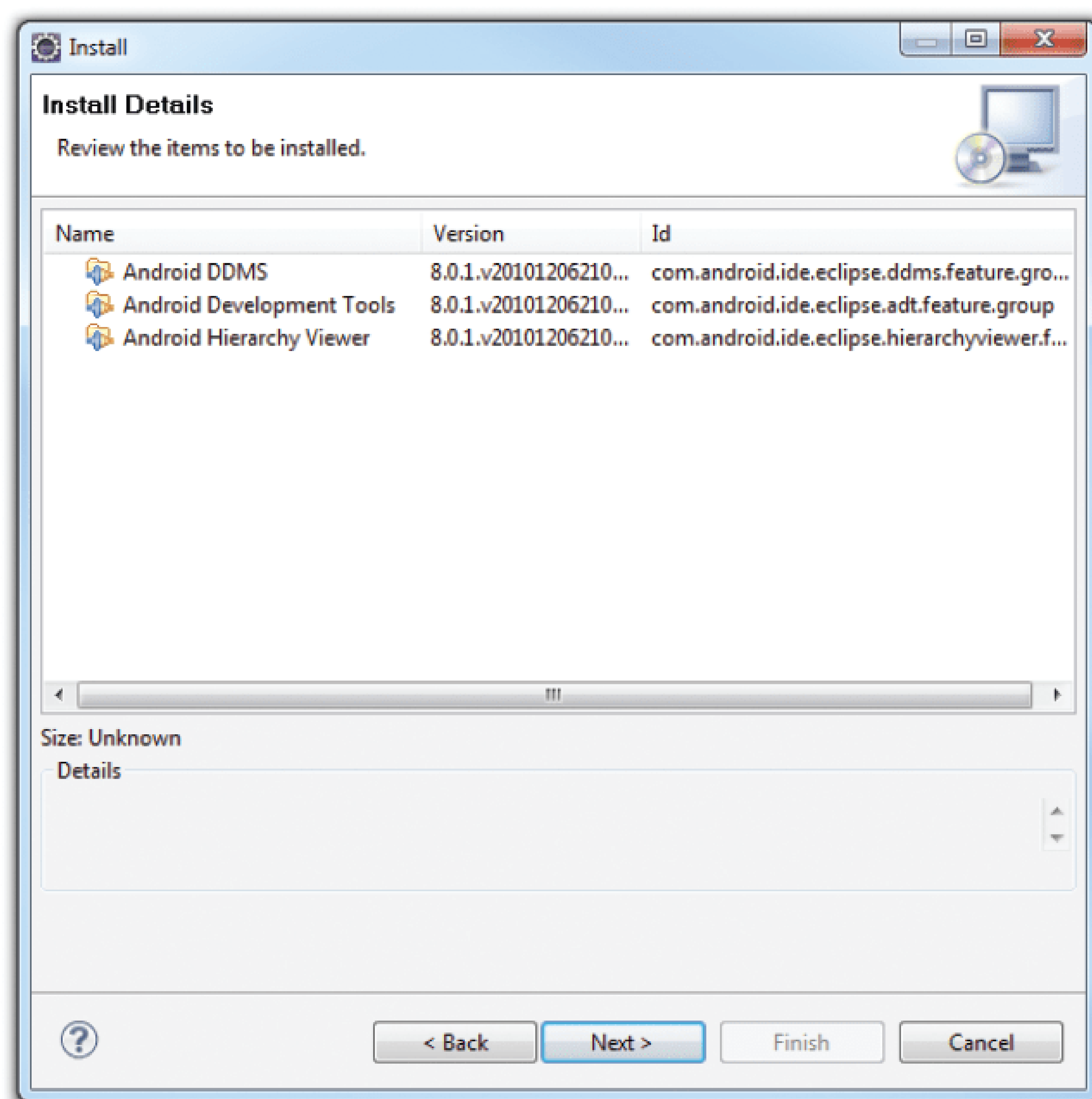


图 1-11

您会被要求查看工具的许可证，选中接受许可协议的选项(如图1-12所示)。单击Finish按钮以继续。

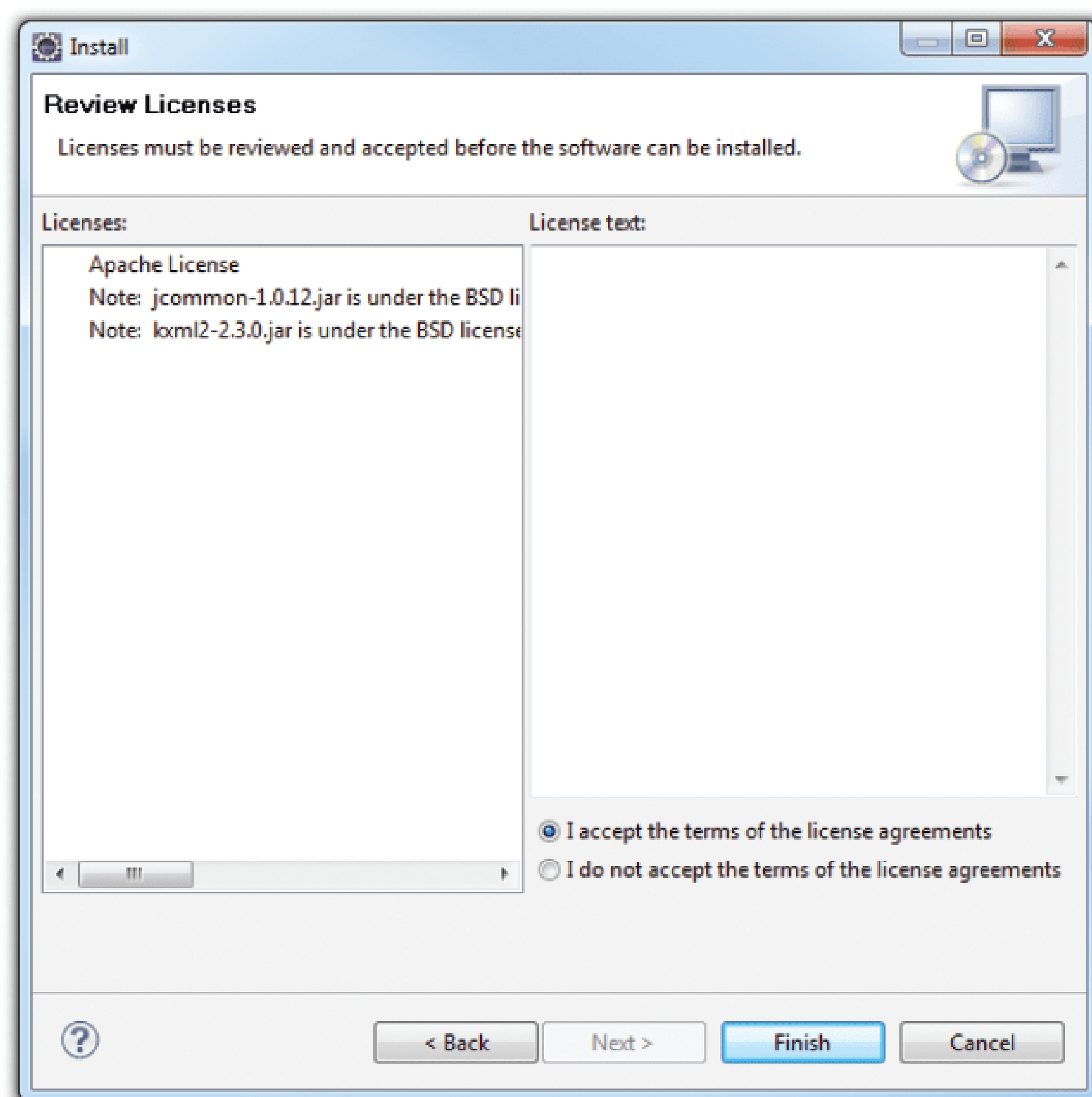
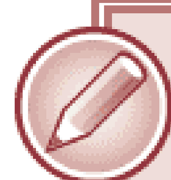


图 1-12

Eclipse将开始从Internet上下载工具并进行安装(如图1-13所示)，这将花点时间，请耐心等待。



注意： 如果在下载ADT过程中遇到任何问题，可以在http://developer.android.com/sdk/eclipse-adt.html#installing上查找Google的帮助。

ADT安装完毕后，将会提示您重启Eclipse。重启后选择Window | Preferences(如图1-14所示)。

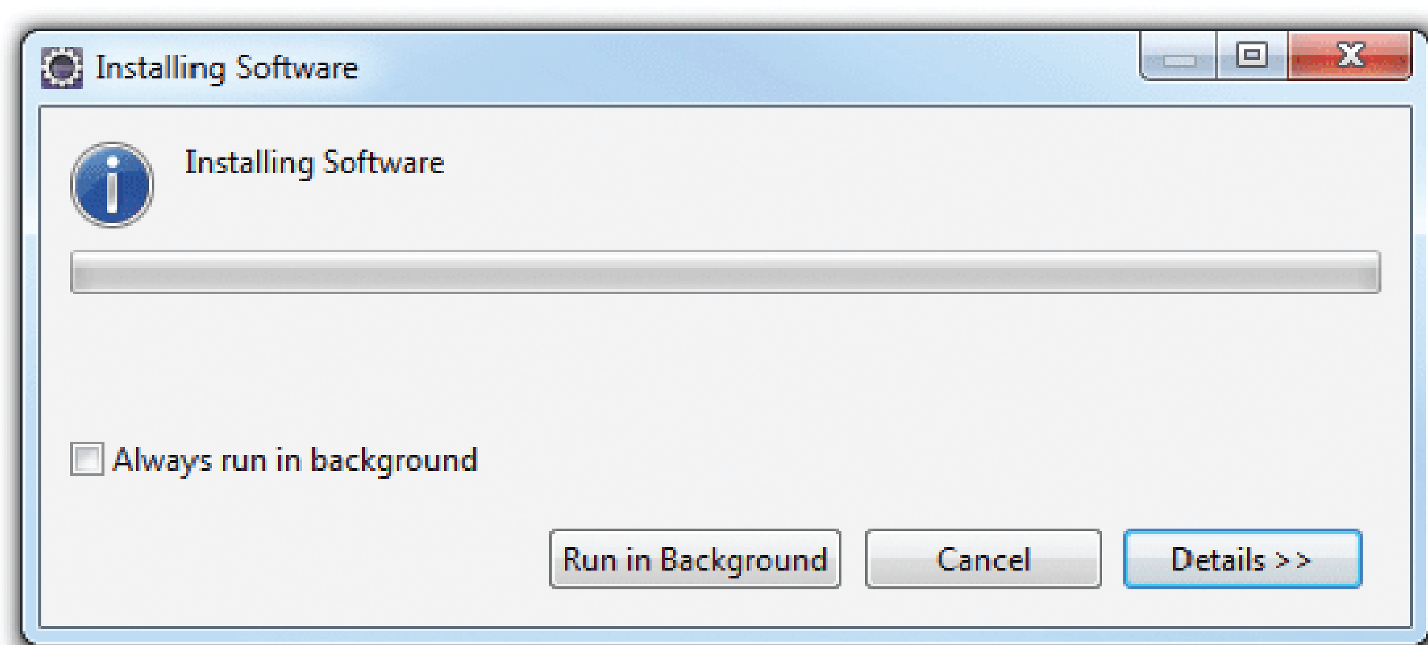


图 1-13

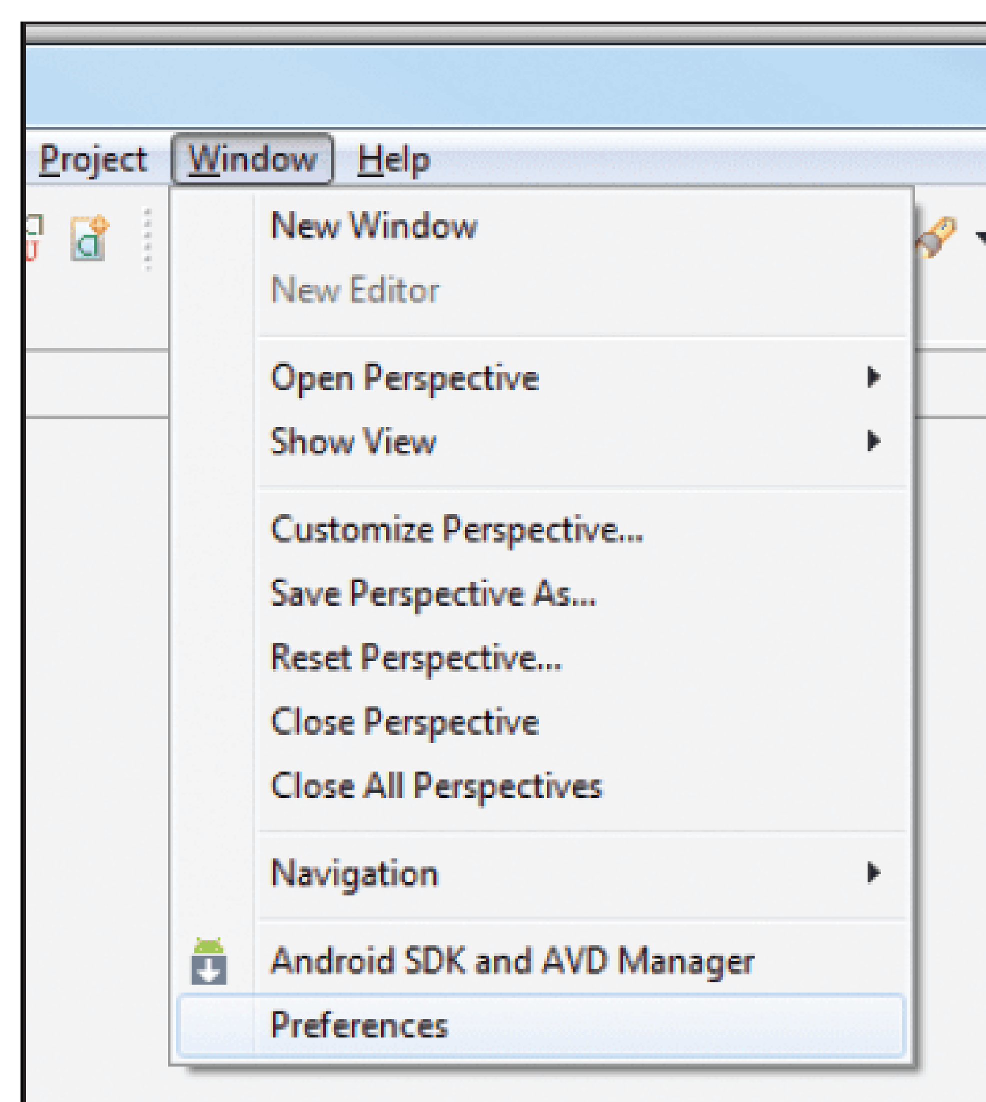


图 1-14

在Preferences窗口中选择Android。这时您将会看到有个错误消息说SDK尚未安装(如图1-15所示)，单击OK按钮忽略该消息。

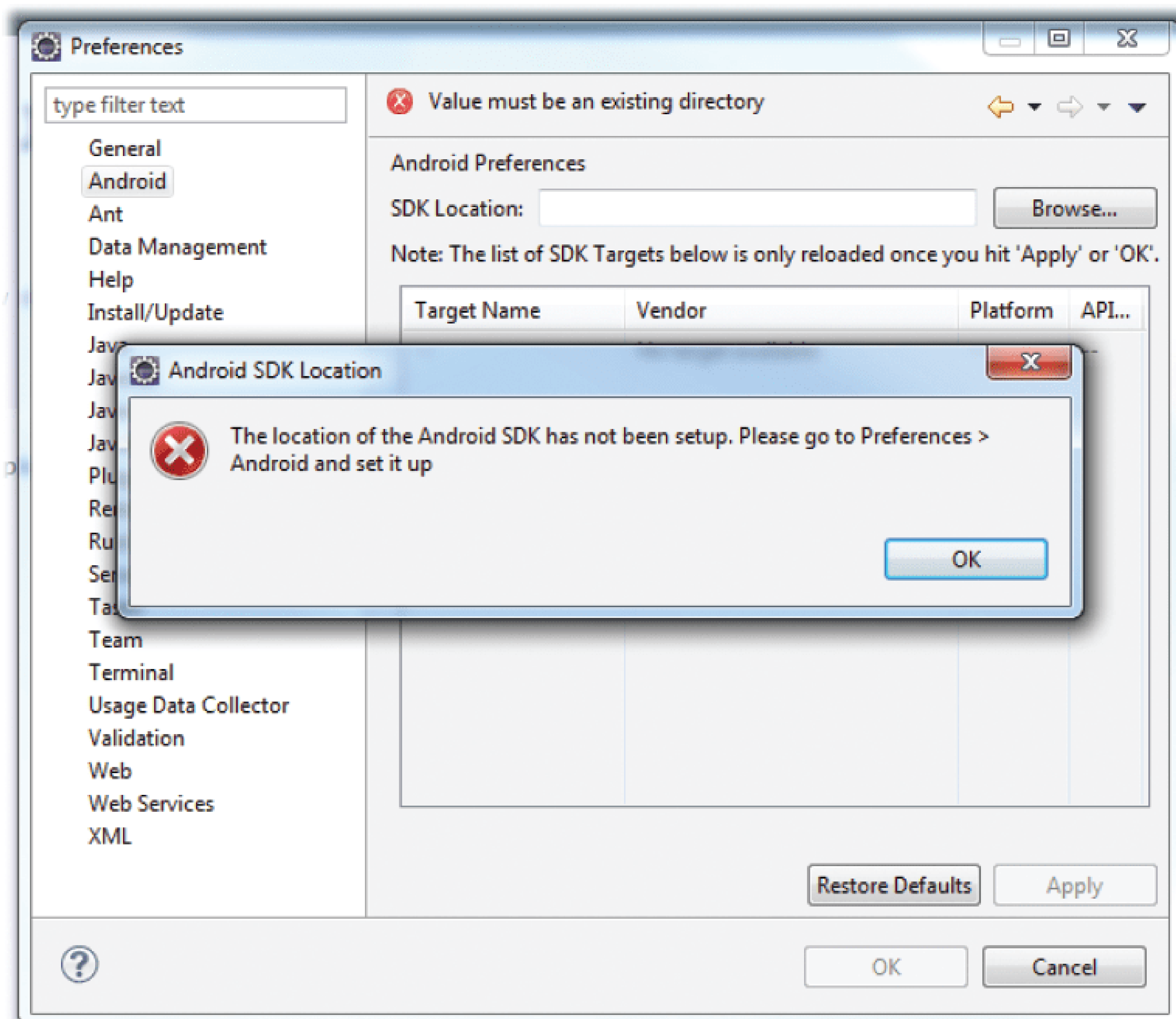


图 1-15

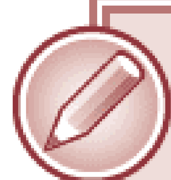
输入Android SDK文件夹的位置，本例中为C:\Android\android-sdk-windows，然后单击OK按钮。

1.2.4 创建Android虚拟设备(AVD)

下一步是创建用于测试Android应用程序的AVD。AVD表示Android虚拟设备(Android Virtual Device)。AVD是一个模拟器实例，可以用来模拟一个真实的设备。

每一个AVD包含一个硬件配置文件、一个到系统映像的映射，以及模拟存储器(例如安全数字(SD)卡)。

您打算测试多少个不同配置的应用程序，就可以创建多少个AVD。这种测试对于确定应用程序在有着不同功能的不同设备上运行时的行为是很重要的。



注意：附录B将讨论Android模拟器的部分功能。

为了创建AVD，选择Window | Android SDK and AVD Manager。

在左边的窗格中选择Available packages选项并在右边的窗格中展开这些包的名称。图1-16显示了可用来创建AVD的不同的包，从而模拟不同版本的Android设备。

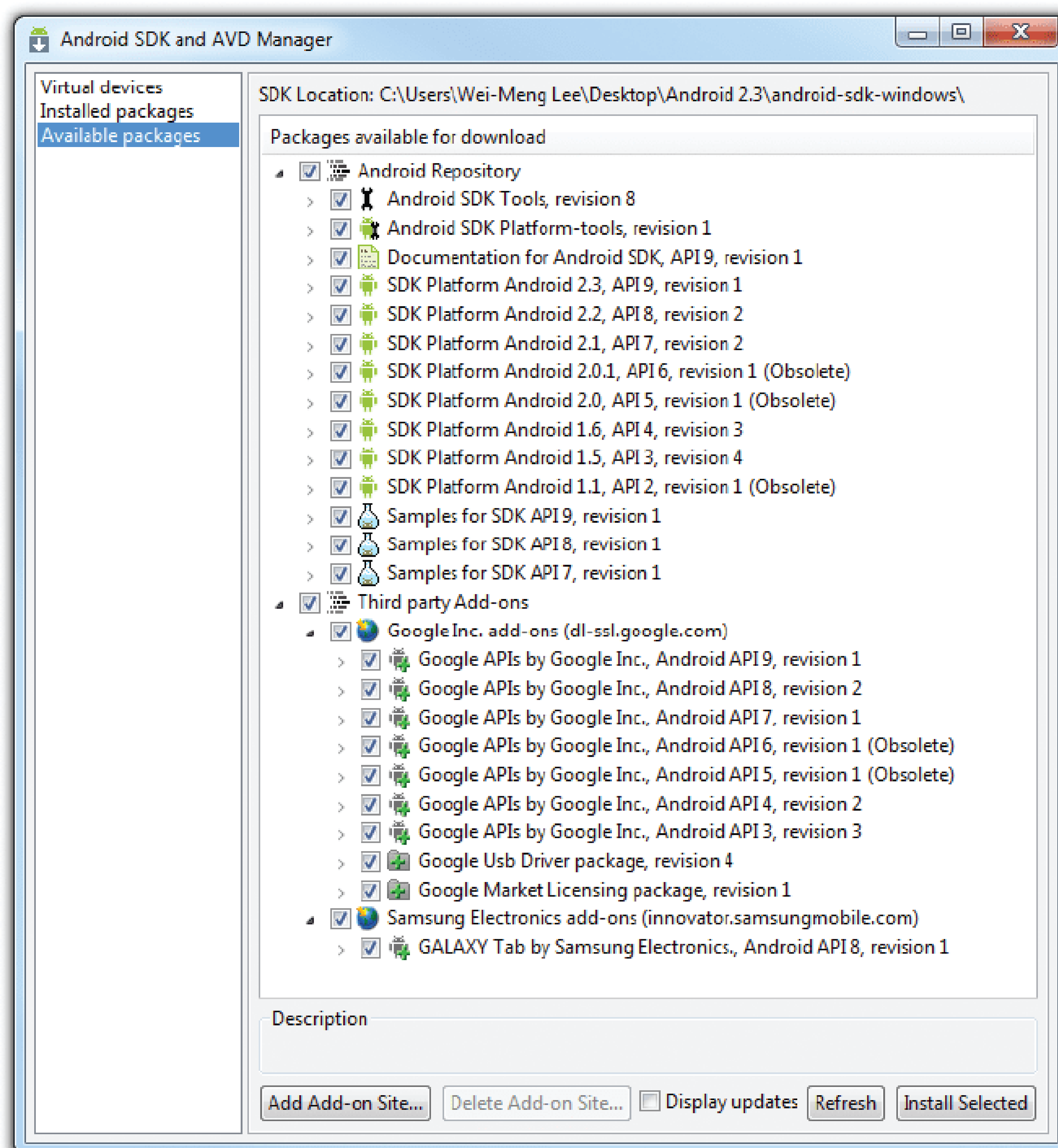


图 1-16

选中您的项目所需的相应工具、文档和平台。

选择好所需的项后，单击Install Selected按钮进行下载。由于从Google的服务器上下载比较费时，因此建议只下载急需的部分，其他的可以等您有时间时再下载。



注意：首先，您至少应该选择最新的SDK平台。在写作本书时，最新的SDK平台是SDK Platform Android 2.3, API 9, revision 1。

每个版本的Android操作系统由一个API级别号来标识。例如，Android 2.3是级别9(API 9)，而Android 2.2是级别8 (API 8)等。对于每一个级别，有两个平台可用。例如，级别9提供了如下两个平台：

- SDK Platform Android 2.3
- Google公司的Google APIs

以上两者的关键区别在于Google APIs平台包含了Google Maps库。因此，如果您在编写需要Google Maps的应用程序，那么需要创建一个使用Google APIs平台的AVD(更多内容请参阅第9章“基于位置的服务”)。

单击窗口左窗格中的Virtual devices项。再单击位于窗口右窗格中的New...按钮。

在Create new Android Virtual Device (AVD)窗口中，输入如图1-17所示的各项内容。完成后单击Create AVD按钮。

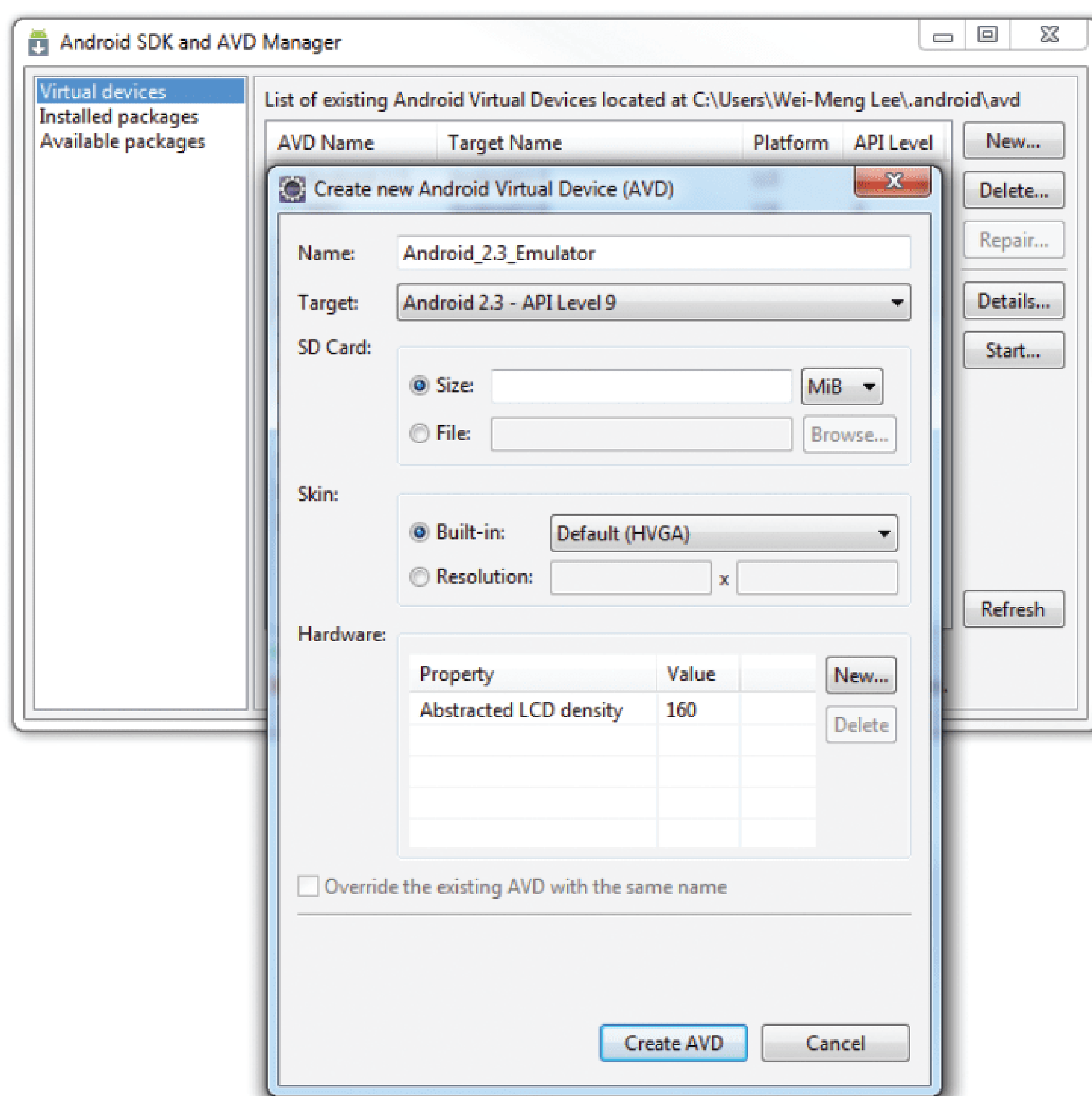
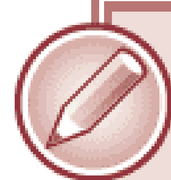


图 1-17

在这里，您已经创建了一个AVD(简言之，一个Android模拟器)，可以用来模拟运行2.3版本操作系统的Android设备。除了所创建的AVD之外，还可以选择模拟具有SD卡和不同屏幕像素密度和分辨率的设备。



注意： 附录B介绍了如何模拟不同类型的Android设备。

创建一些具有不同API级别的AVD会更好些，这样您的应用程序可以在不同设备上得到测试。图1-18给出了这样一个示例，通过创建多个AVD，从而在大量不同的Android平台上测试您的应用程序。

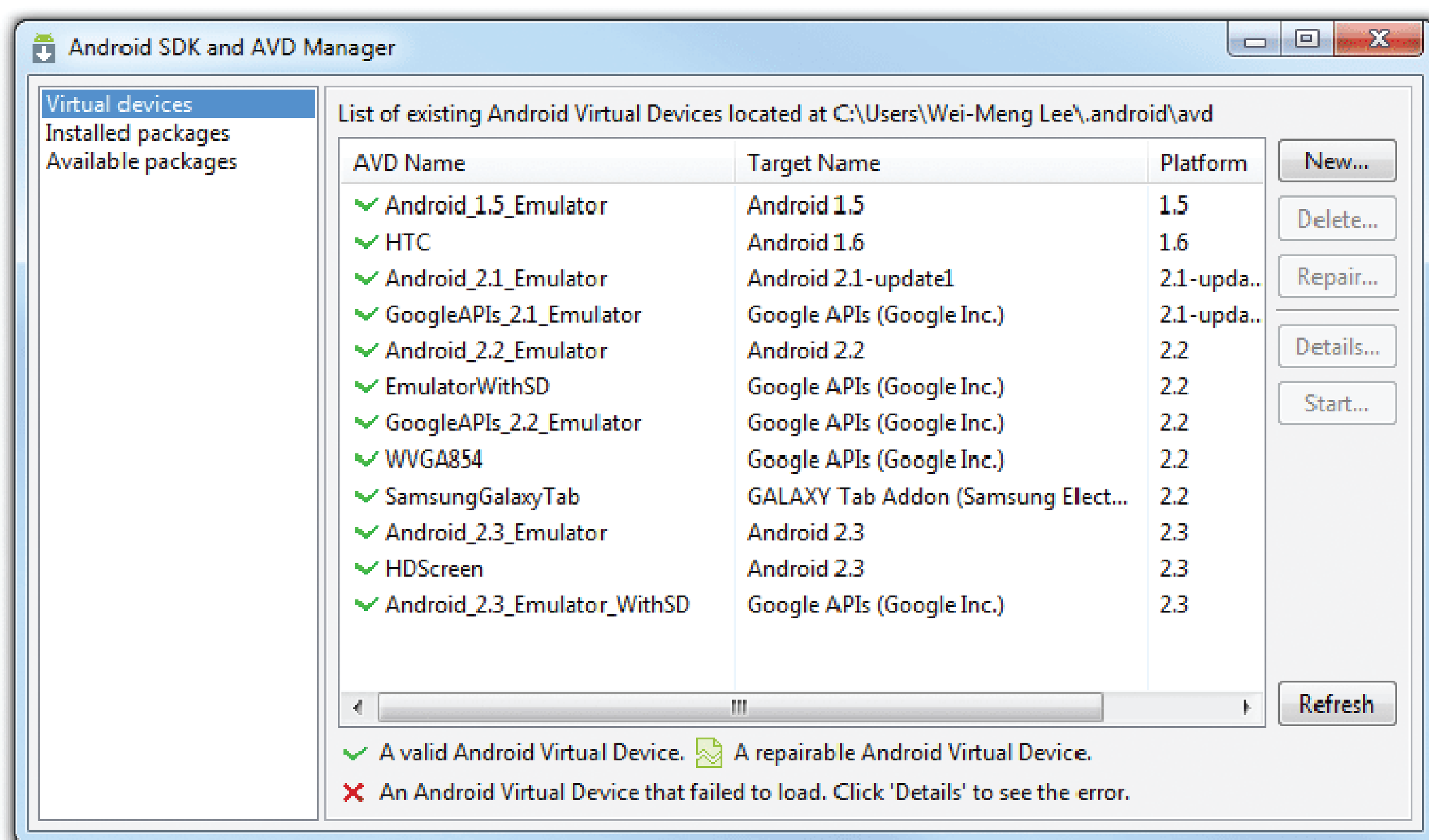


图 1-18

1.2.5 创建第一个Android应用程序

在所有的工具和SDK都下载和安装好以后，现在是开动马达的时候了。和所有的编程书籍一样，第一个示例是用无所不在的Hello World应用程序。这将有助于您详细了解构成一个Android项目的不同组件。

闲话少叙，让我们直接进入Android的世界！

试一试 创建第一个Android应用程序

HelloWorld.zip代码文件可以在Wrox.com上下载

(1) 启动Eclipse，选择菜单File | Project...创建一个新项目(如图1-19所示)。

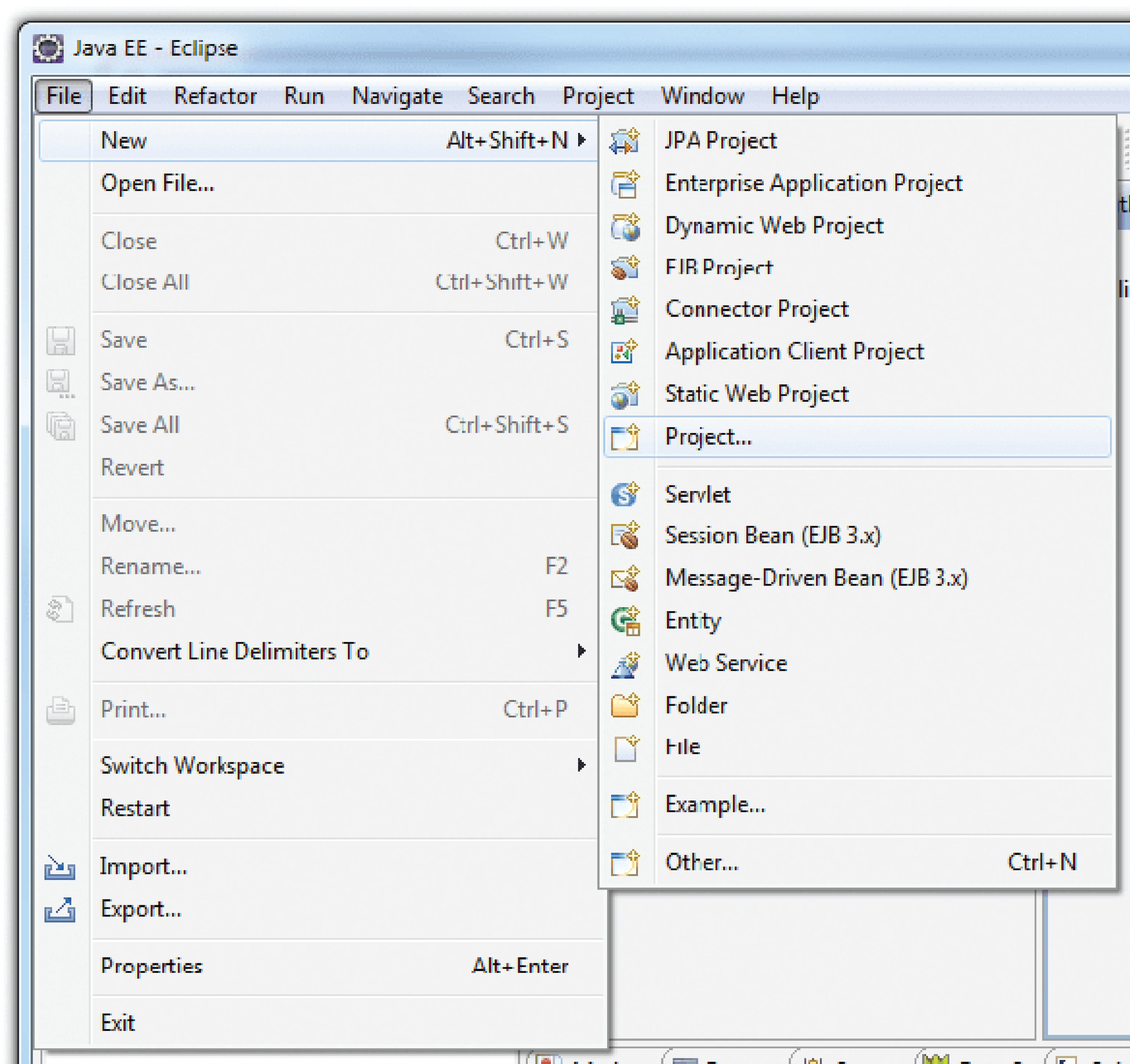
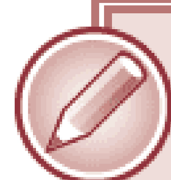


图 1-19



注意：在创建了您的第一个Android应用程序后，以后的Android项目可以通过依次选择菜单项File | New | Android Project来创建。

(2) 展开Android文件夹，选择Android Project(如图1-20所示)。

(3) 按图1-21所示为Android项目命名，然后单击Finish按钮。

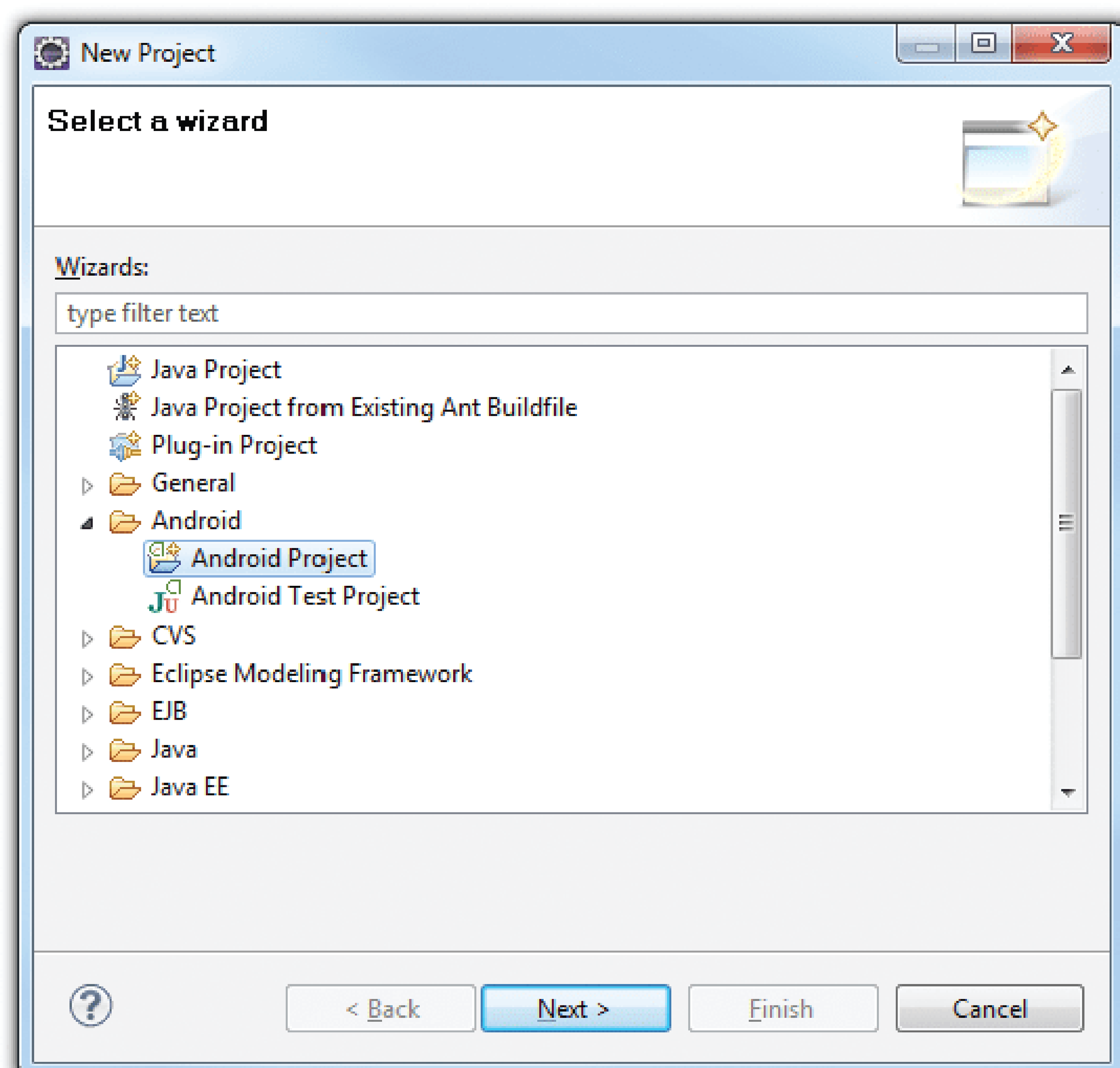


图 1-20

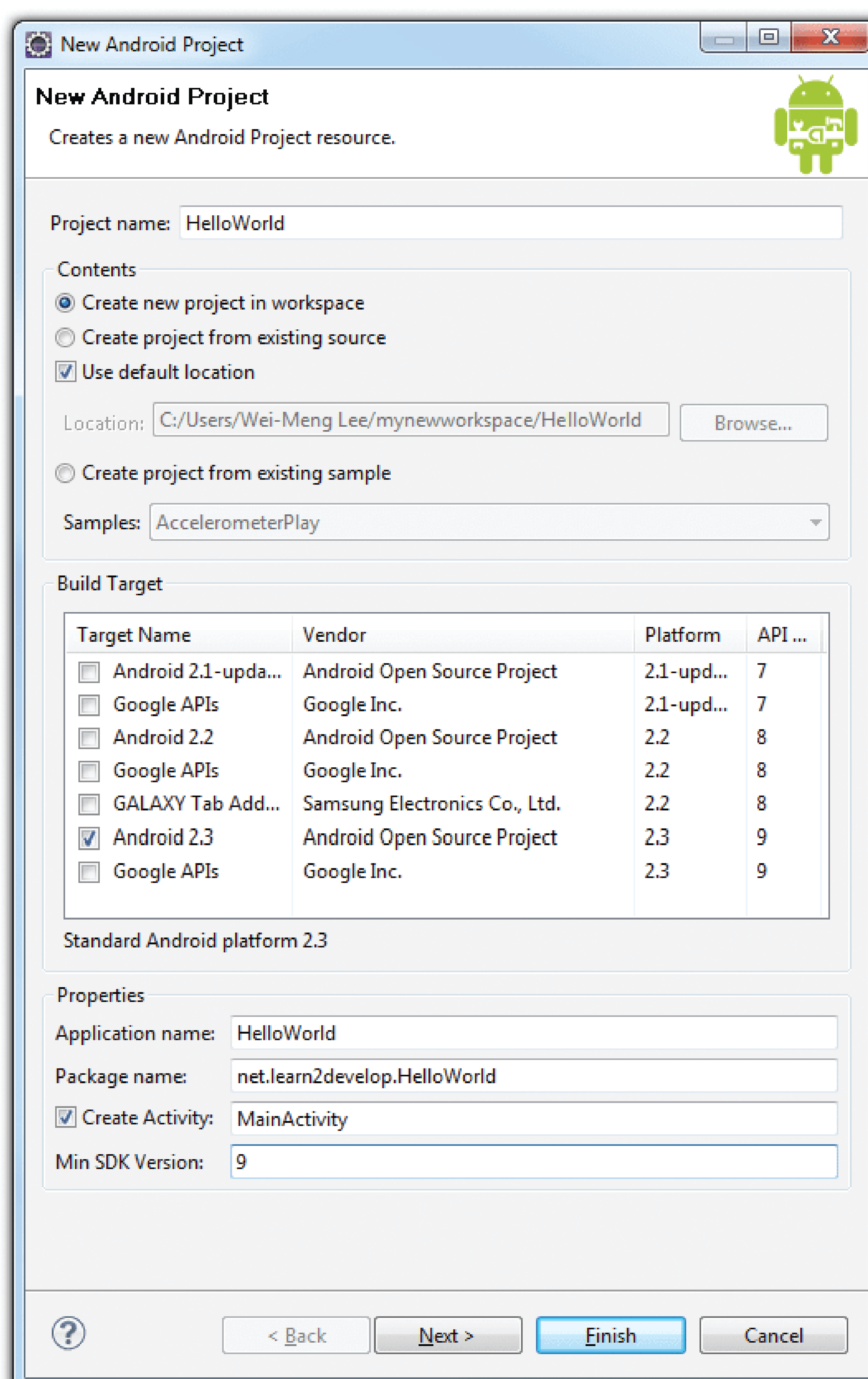
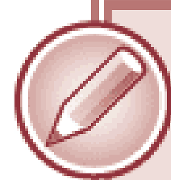


图 1-21



注意：您要在包的名称中至少包含一个句点(.)。包名称的惯例是使用反向域名，项目名称紧随其后。例如，我公司的域名是learn2develop.net，因此我的包的名称应该是net.learn2develop.HelloWorld。

(4) 此时，Eclipse IDE应该如图1-22所示。

(5) 在Package Explorer窗口(位于Eclipse IDE的左边)中，单击项目中每个项左侧显示的各种箭头，展开HelloWorld项目。在res/layout文件夹中，双击main.xml文件(如图1-23所示)。

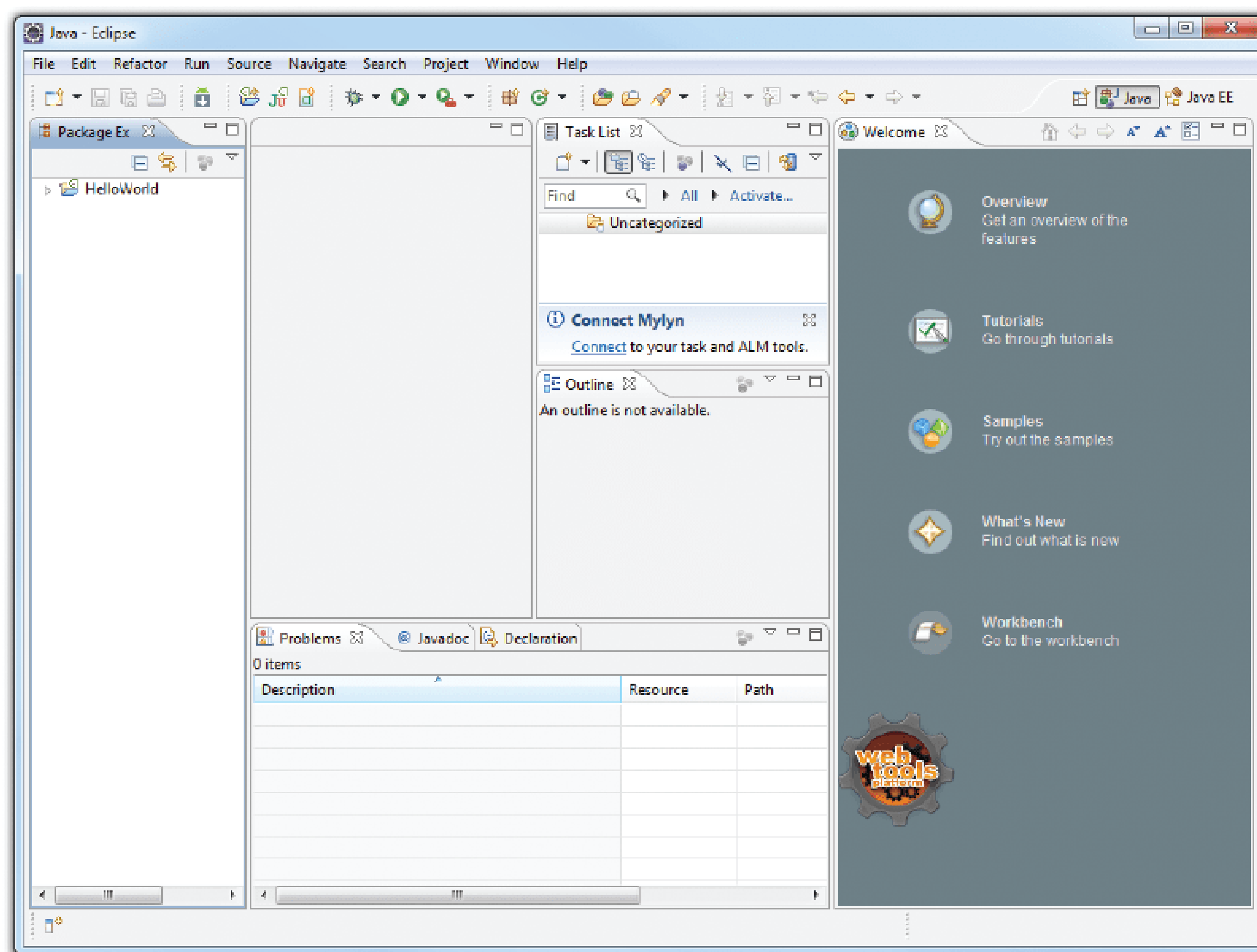


图 1-22

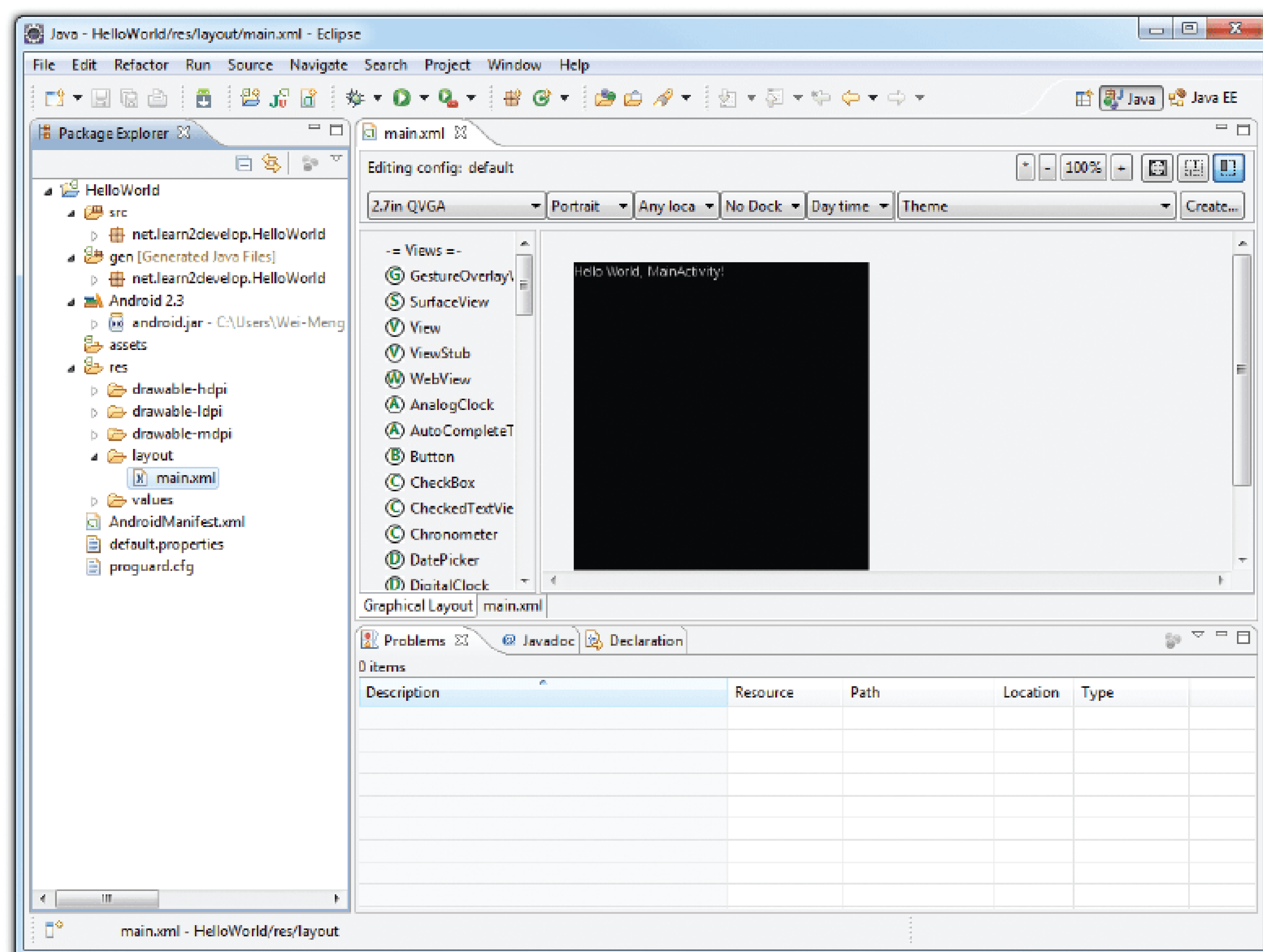


图 1-23

(6) main.xml文件定义了应用程序的用户界面(UI)。默认视图是Layout视图，以图形化的方式显示了活动。要修改该用户界面，可单击位于底部的main.xml选项卡(如图1-24所示)。

(7) 把下列粗体显示的代码添加到main.xml文件中：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
```

```
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="This is my first Android Application!" />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="And this is a clickable button!" />

    </LinearLayout>
```

(8) 按Ctrl+s组合键保存对项目的修改。

(9) 现在可以着手准备在Android模拟器上测试应用程序了。在Eclipse中选择项目名称并按F11键。系统将要求您选择一种方法来调试应用程序。选择如图1-25所示的Android Application，并单击OK按钮。

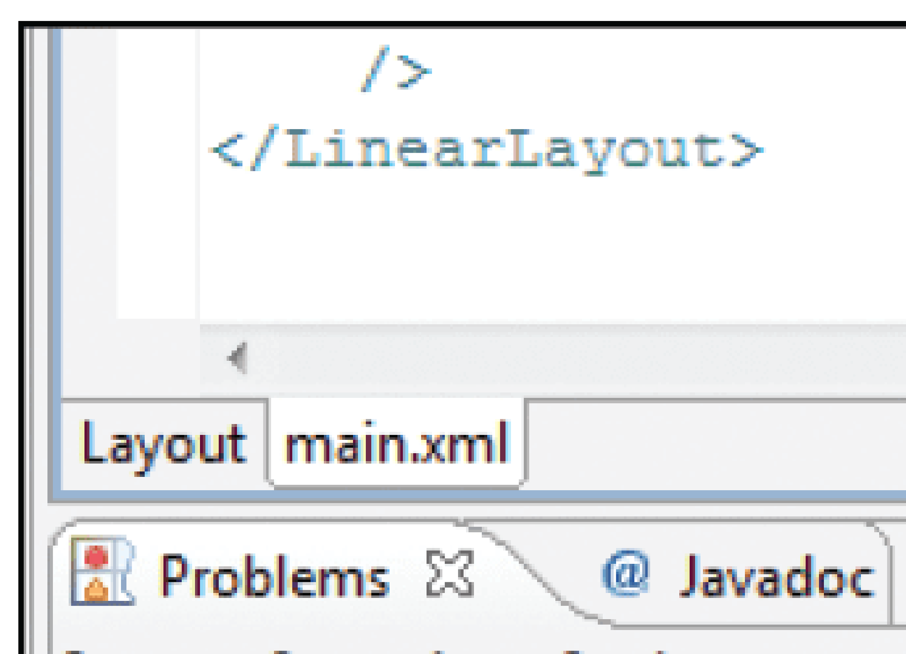


图 1-24

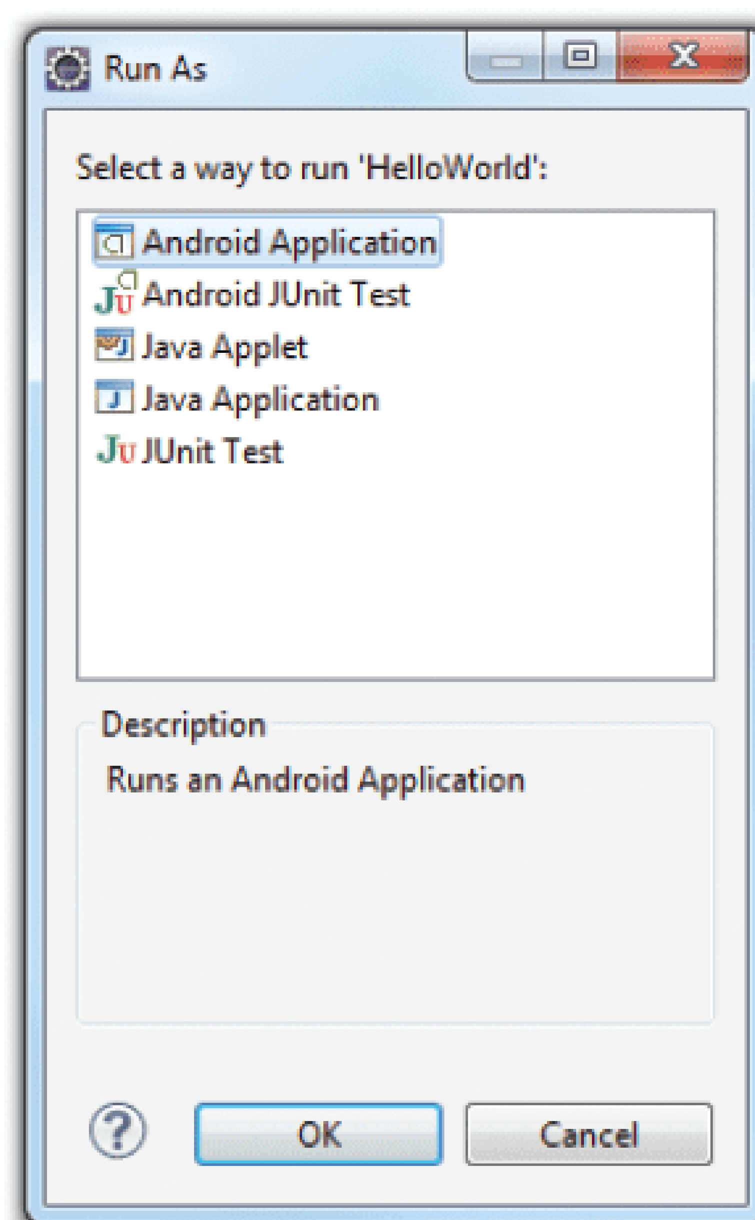
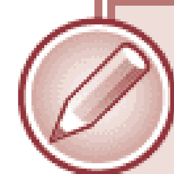


图 1-25



注意：有些Eclipse安装有个很讨厌的错误：在创建一个新项目后，当您想调试应用程序时，Eclipse报告说项目中包含错误。甚至在您没有修改项目的任何文件或文件夹时也会如此。为了解决这一问题，可以直接删除位于gen/net.learn2develop.HelloWorld文件夹下的R.java文件。Eclipse将会为您自动生成一个新的R.java文件。一旦做完这一步，项目将不再包含任何错误。

(10) 现在Android模拟器将开始启动(如果模拟器被锁住，那么需要首先滑动解锁按钮解锁)。图1-26显示了运行于Android模拟器上的应用程序。

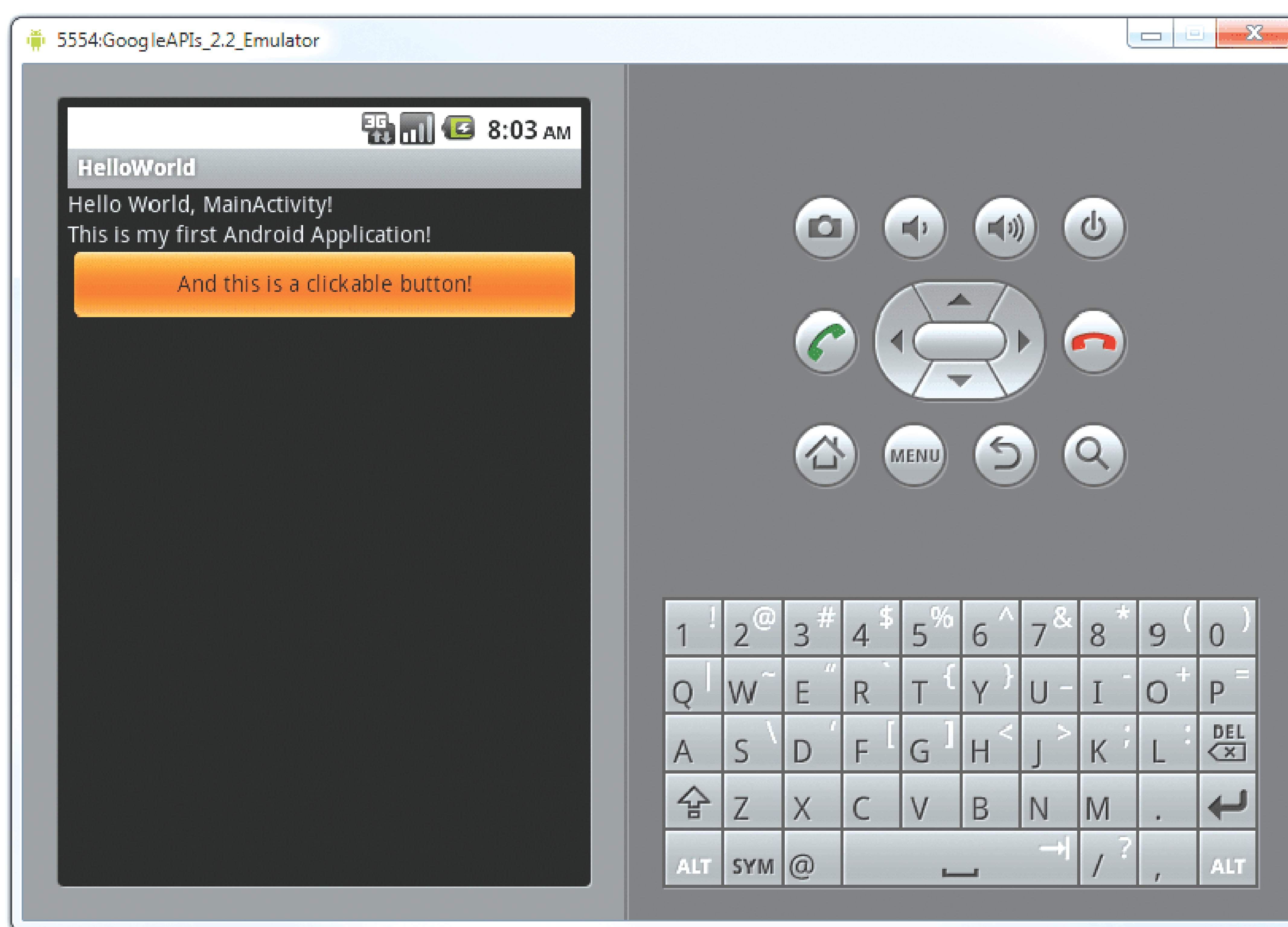


图 1-26

(11) 单击Home按钮(位于键盘上左下角的房子图标)，将显示主屏内容(如图1-27所示)。



图 1-27

(12) 单击应用程序的启动器图标来显示已安装到设备上的应用程序列表。注意，HelloWorld现在已安装在应用程序启动器中(如图1-28所示)。



图 1-28

将用哪一个AVD来测试您的应用程序

回忆一下先前使用AVD Manager创建的几个AVD。那么，当运行一个Android应用程序时，Eclipse将启动哪一个呢？Eclipse会检查您(当创建一个新项目时)指定的目标，将其与已经创建好的AVD列表对照，然后启动第一个匹配的AVD来运行您的应用程序。

如果在调试应用程序前有多个合适的AVD正在运行，Eclipse将显示Android Device Chooser窗口，使您可以从中选择想要的模拟器或设备来调试应用程序(如图1-29所示)。

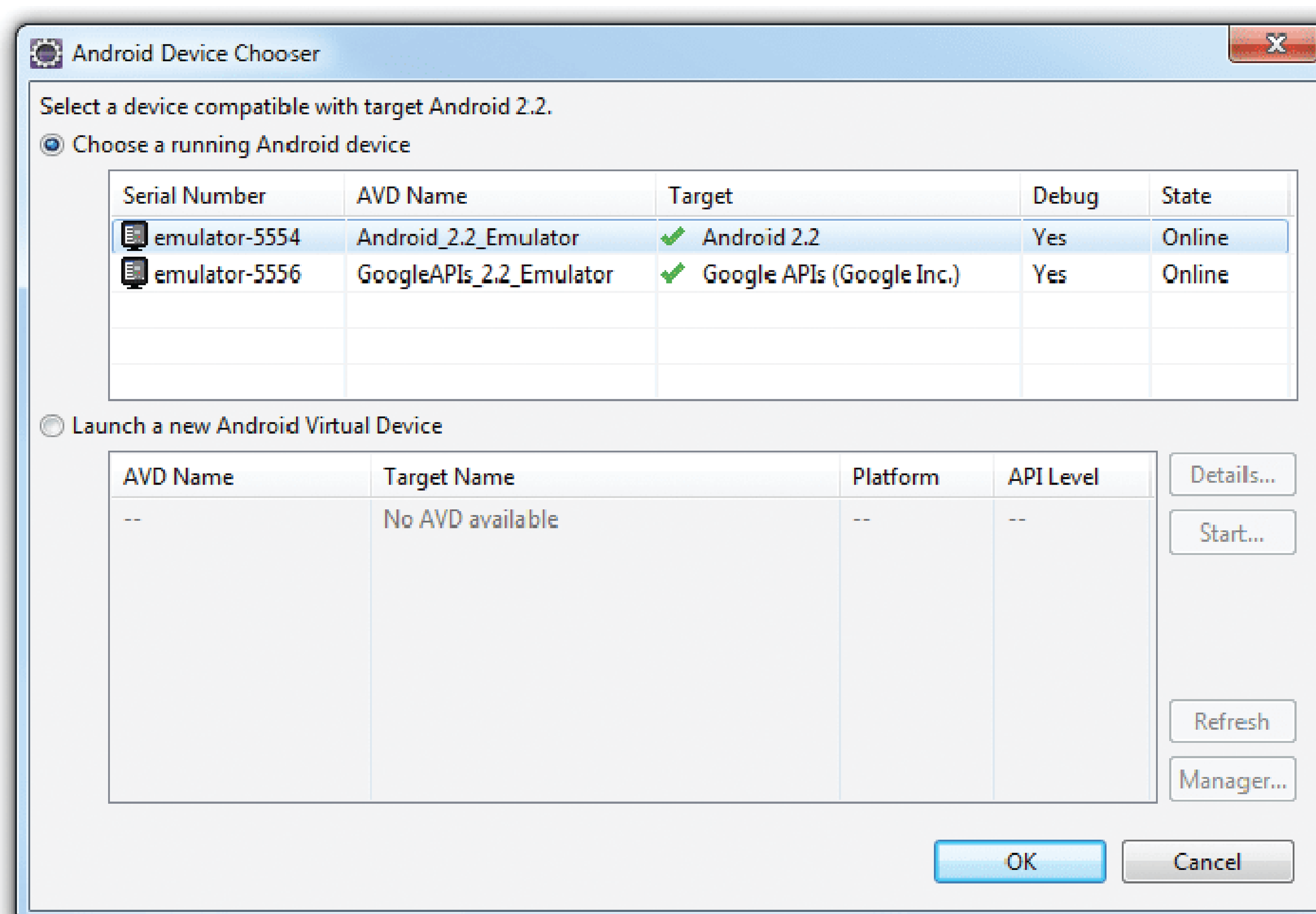


图 1-29

示例说明

为了使用Eclipse创建一个Android项目，需要提供如表1-2所示的信息。

表1-2　默认方式创建的项目文件

属　　性	描　　述
Project name	项目的名称
Application name	用户友好的应用程序名称
Package name	包的名称，必须使用反向域名
Create Activity	应用程序中第一个活动的名称
Min SDK Version	项目所需的最低版本的SDK

在Android中，Activity(活动)是一个包含您的应用程序的用户界面的窗口。一个应用程序可以有零或多个活动。本例中，应用程序包含一个活动：MainActivity。这个MainActivity是应用程序的入口点，在应用程序启动时显示。第2章将详细讨论活动。

在这个简单的示例中，修改main.xml文件以显示字符串“This is my first Android Application!”和一个按钮。main.xml文件包含了这个活动的用户界面，此界面在MainActivity加载时显示。

当在Android模拟器上调试应用程序时，应用程序会自动在模拟器上进行安装——没错，您已经开发了您的第一个Android应用程序！

1.2.6节将揭示您的Android项目中所有这些不同的文件是如何一起工作来使程序正常运行的。

1.2.6　Android应用程序剖析

既然已经创建了您的第一个Hello World Android应用程序，那就该分析一下Android项目的内部结构，检查一下使之工作的所有部件。

首先，请注意在Eclipse的Package Explorer中所显示的构成Android项目的不同文件(如图1-30所示)。

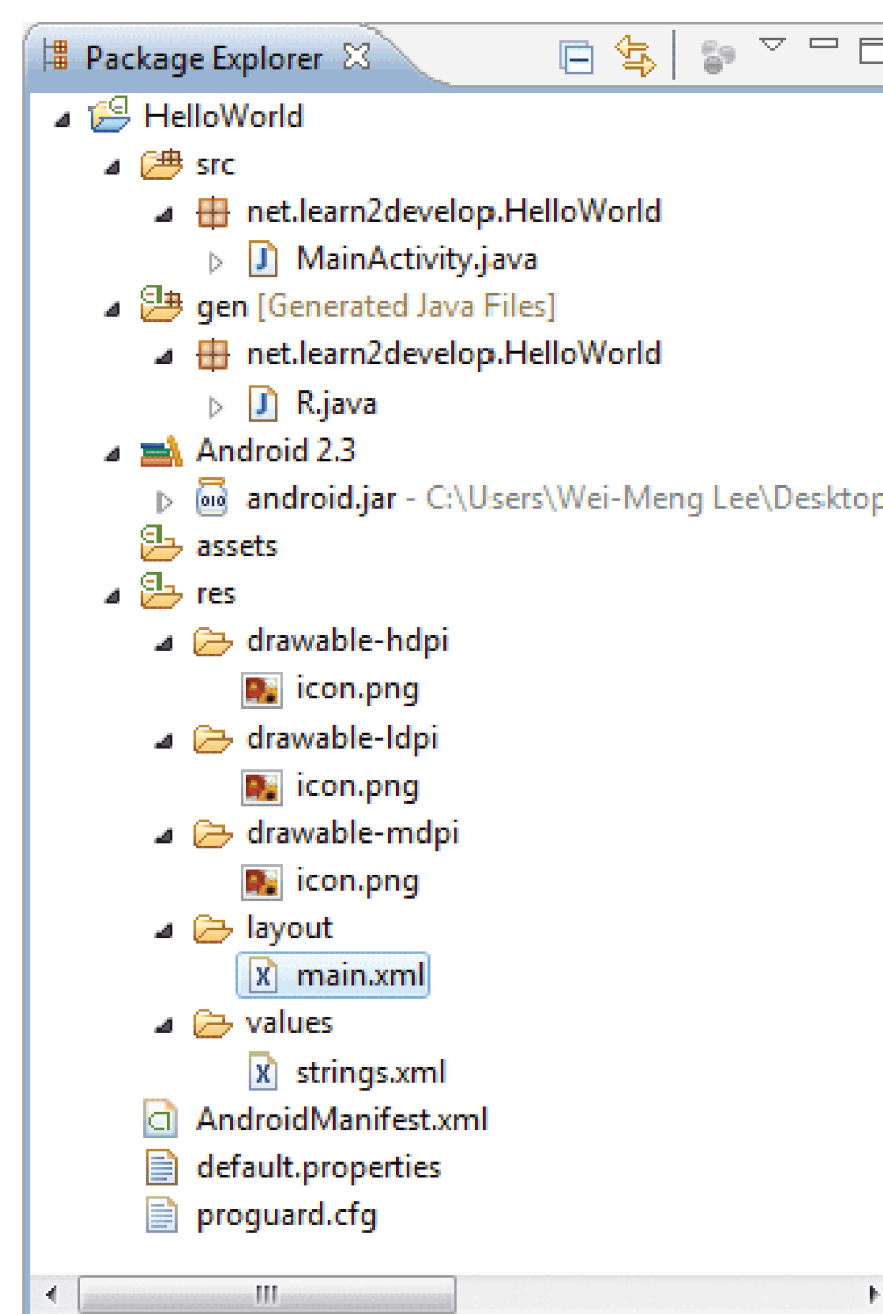


图 1-30

这些不同的文件夹及其文件如下：

- src——包含项目的.java源文件。在本例中，有一个文件：MainActivity.java。MainActivity.java文件是活动的源文件，您将在这个文件中编写应用程序的代码。
- Android 2.3库——这一项中有一个android.jar文件，包含了一个Android应用程序所需的所有类库。
- gen——包含了由编译器生成的R.java文件，它引用在项目中能找到的全部资源。不要修改此文件。
- assets——这个文件夹包含了应用程序所用到的所有资产，例如HTML、文本文件、数据库等。
- res——这个文件夹包含了应用程序中使用的所有资源。它还包含了几个子文件夹：drawable-<*resolution*>、layout和values。第3章将进一步讨论如何支持具有不同屏幕分辨率和像素密度的设备。
- AndroidManifest.xml——这是Android应用程序的清单文件。在这一文件中，可以指定应

用程序所需的权限，还可以指定其他特性(如意图筛选器、接收者等)。第2章将详细讨论AndroidManifest.xml文件的使用。

main.xml文件定义了活动的用户界面。注意观察以下代码的粗体字部分：

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

这里，@string指的是位于res/values文件夹下的string.xml文件。因此，@string/hello指的是在strings.xml文件中定义的hello字符串，即“Hello World, MainActivity!”：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, MainActivity!</string>
    <string name="app_name">HelloWorld</string>
</resources>
```

建议您将应用程序中所有的字符串常量存储于这个string.xml文件中，并用@string标识符引用这些字符串。这样，如果需要将您的应用程序本地化为另一种语言，则只需要用目标语言替换string.xml文件中存储的字符串，然后再把应用程序重新编译一遍。

观察一下AndroidManifest.xml文件的内容：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.HelloWorld"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
</manifest>
```

文件AndroidManifest.xml包含了关于应用程序的详细信息。

- 它定义了应用程序的包名：net.learn2develop.HelloWorld。
- 应用程序的版本代码为1。这个值是用来标识您的应用程序的版本号。它可用于以编程方式确定应用程序是否需要升级。
- 应用程序的版本名称是1.0。此字符串值主要用来显示给用户。这个值应该采用以下格式：*<major>.<minor>.<point>*。
- 应用程序使用位于drawable文件夹下的图像icon.png。
- 应用程序的名称是在string.xml文件中定义的名为app_name的字符串。

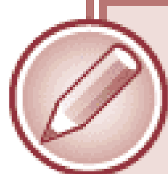
- MainActivity.java文件代表了应用程序中的一项活动。代表这项活动的标签名称与应用程序的名称相同。
- 在这项活动的定义中，有一个名为<intent-filter>的元素:
  - 意图筛选器的动作名称为android.intent.action.MAIN，表明了这项活动是应用程序的入口点。
  - 意图筛选器的类别名称为android.intent.category.LAUNCHER，表明了应用程序可从设备的启动器图标启动。第2章将详细讨论意图。
- 最后，<uses-sdk>元素的android:minSdkVersion属性指定了应用程序运行所需的操作系统的最低版本。

在向项目中加入更多的文件和文件夹后，Eclipse将自动生成R.java的内容，而目前包含以下内容：

```
package net.learn2develop.HelloWorld;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

建议您不要修改R.java文件的内容。当您修改项目时，Eclipse会自动为您生成相应内容。



注意：如果手动删除了R.java文件，Eclipse会为您立即再重新生成一个。注意，为了使Eclipse可以生成R.java文件，您的项目不能包含任何错误。如果在删除R.java后发觉Eclipse没有重新生成这个文件，那么您需要再检查一遍您的项目。代码中可能包含语法错误或者XML文件(如AndroidManifest.xml、main.xml等)的格式不良好。

最后，把活动连接到用户界面(main.xml)的代码是位于MainActivity.java文件中的setContentView()方法：

```
package net.learn2develop.HelloWorld;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {
```

```
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

这里，R.layout.main指的是位于res/layout文件夹下的main.xml文件。在向res/layout文件夹添加额外的XML文件时，R.java中将自动生成这个文件名。onCreate()方法是活动加载时被调用的多个方法之一。第2章将详细讨论活动的生命周期。

1.3 本章小结

本章介绍了Android的概况，并强调了它的一些功能。如果您已经按照本章前面所述下载了工具和SDK，那么现在应该有了一个工作系统——一个能够开发其他比Hello World更有趣的Android应用程序的系统。在第2章，您将学习到有关活动和意图的概念以及这些概念在Android中所扮演的重要角色。

练 习

1. 什么是AVD?
2. AndroidManifest.xml文件中的android:versionCode和android:versionName属性有什么区别?
3. strings.xml文件的作用是什么?

练习答案参见附录C。

本章主要内容

主 题	关 键 概 念
Android操作系统	Android是一个基于Linux的开源的手机操作系统。它可以供任何打算使之在其自己设备上运行的用户使用
Android应用开发语言	使用Java编程语言开发Android应用程序。编写的应用程序被编译成可在Dalvik虚拟机之上运行的Dalvik可执行文件
Android Market	Android Market包括了由第三方开发人员编写的各种Android应用程序
Android应用程序开发工具	Eclipse IDE、Android SDK和ADT
活动	Android应用程序中的一个屏幕代表一个活动。每一个应用程序可以有零或多个活动
Android清单文件	AndroidMainifest.xml文件包含了应用程序的详细配置信息。随着应用程序变得更加复杂，您需要不断修改这个文件，同时在本书的学习过程中，您将看到可添加到这个文件中的不同信息

第2章 活动和意图

本章将介绍以下内容

- 什么是活动
- 如何对活动应用样式和主题
- 如何将活动显示为对话框窗口
- 理解意图的概念
- 如何使用Intent对象链接活动
- 意图筛选器如何使您有选择地链接到其他活动
- 如何使用通知向用户显示警报

在第1章中，您已经知道了活动就是一个包含应用程序的用户界面的窗口。一个应用程序可以包含零个或多个活动。通常，应用程序具有一个或多个活动，活动的主要目的就是与用户交互。一个活动的生命周期是指从在屏幕上显示那一刻起一直到最后隐藏所经历的若干个阶段。理解活动的生命周期对确保应用程序正确地工作是极其关键的。本章将学习更多有关活动是如何运行以及在设计Android应用程序时必须注意什么的内容。

除了活动，Android中的另一个独特的概念就是意图。一个意图从根本上来说就是能够将来自不同应用程序的不同活动无缝连接在一起工作的“胶水”，确保这些任务执行起来像是都属于一个单一的应用程序。在本章第二部分，您将学习到有关这一重要概念的更多内容，并学会如何使用它来调用诸如Browser、Phone、Maps等内置应用程序。

2.1 理解活动

首先让我们看看是如何创建一个活动的。要创建一个活动，需要创建一个扩展Activity基类的Java类：

```
package net.learn2develop.Activities;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
```

```
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
        }
    }
```

随后，您自己的活动类将使用在res/layout文件夹下定义的XML文件加载此活动的用户界面(UI)组件。本例中，将使用main.xml文件来加载用户界面：

```
            setContentView(R.layout.main);
```

应用程序中的每一个活动必须在AndroidManifest.xml文件中声明，如下所示：

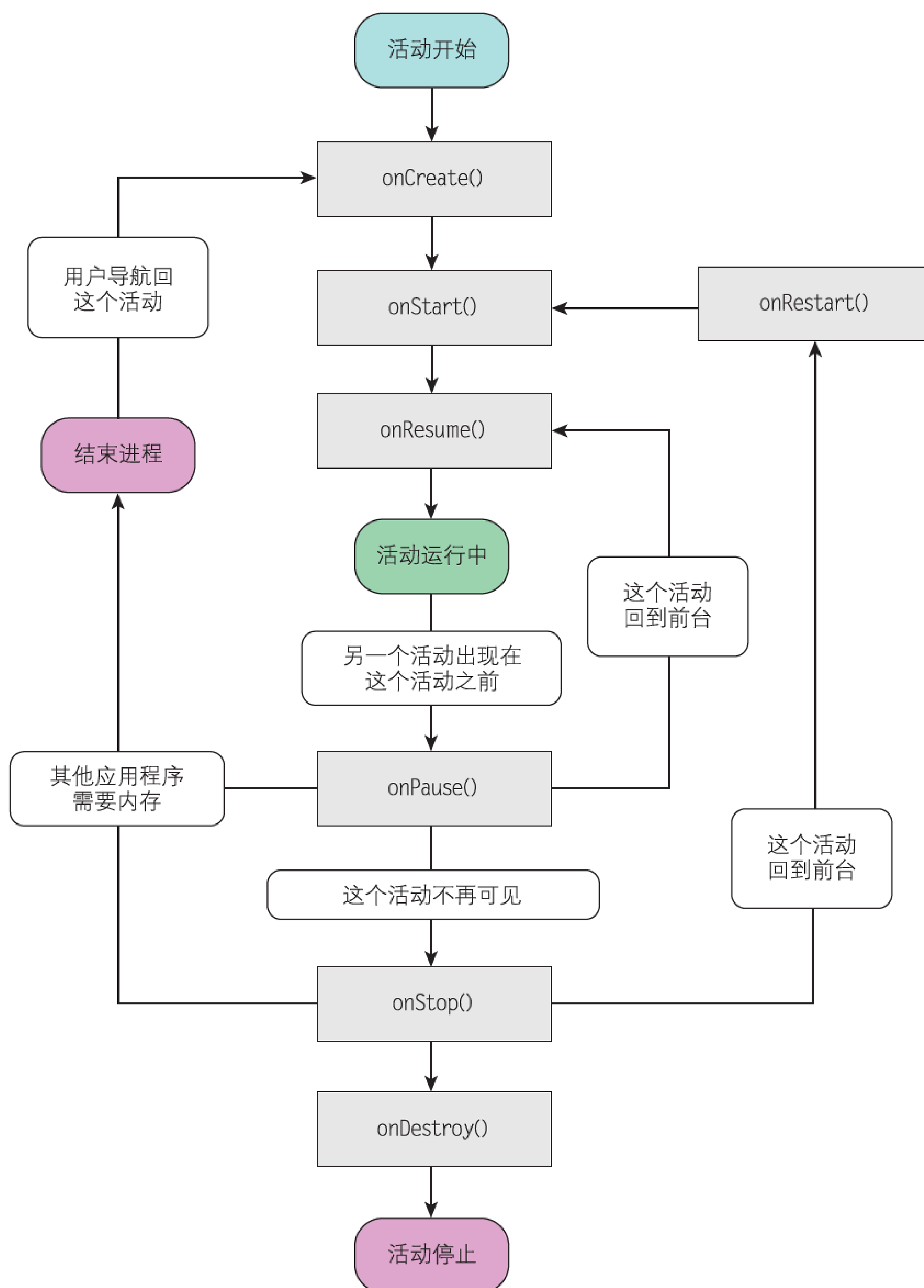
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Activities"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
</manifest>
```

Activity基类定义了管理一个活动的生命周期的一系列事件。该类定义了如下事件：

- onCreate()——当活动首次被创建时调用
- onStart()——当活动对用户可见时调用
- onResume()——当活动与用户开始交互时调用
- onPause()——在当前活动被暂停并恢复以前的活动时调用
- onStop()——当活动不再对用户可见时调用
- onDestroy()——在活动被系统销毁前调用(手动或由系统执行以节省内存)
- onRestart()——在活动已停止并要再次启动时调用

默认情况下，所创建的活动包含OnCreate()事件。在这个事件处理程序中含有帮助显示屏幕的用户界面元素的代码。

图2-1展示了一个活动的生命周期及其所经历的各个阶段——从活动开始直到结束。



本图是依据Creative Commons 2.5的署名许可所描述的条款，复制于Android开源项目创建并共享的工作内容，详见http://developer.android.com/reference/android/app/Activity.html

图 2-1

要了解一个活动所经历的各个阶段，最好的办法是创建一个新项目，实现各种事件，然后使活动经受各种用户交互的考验。

试一试 理解一个活动的生命周期

Activities.zip代码文件可以在Wrox.com上下载

(1) 启动Eclipse，创建一个新的Android项目并命名，如图2-2所示。

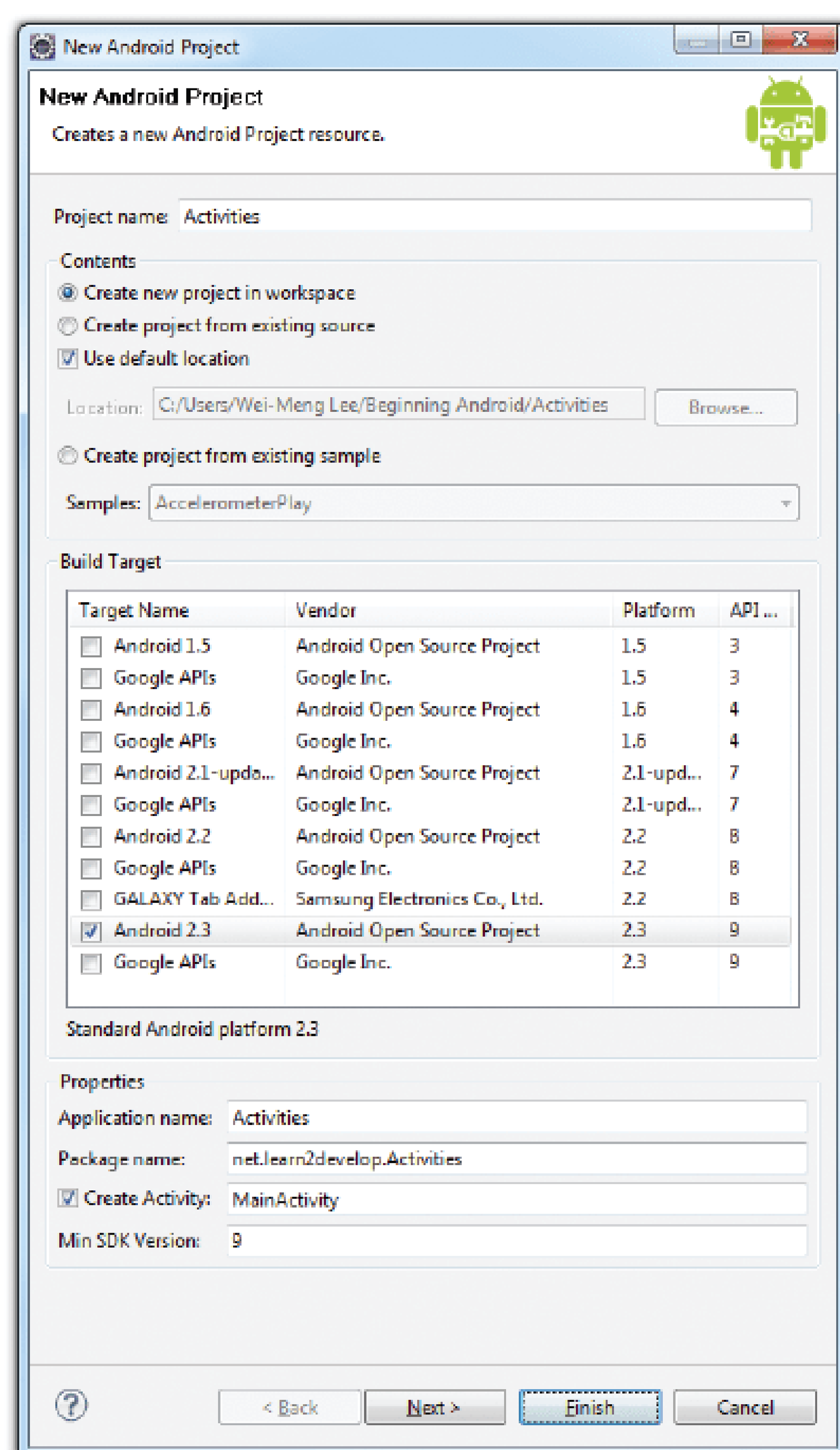


图 2-2

(2) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Activities;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends Activity {
    String tag = "Events";

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Log.d(tag, "In the onCreate() event");
    }
    public void onStart()
    {
        super.onStart();
        Log.d(tag, "In the onStart() event");
    }
    public void onRestart()
    {
        super.onRestart();
        Log.d(tag, "In the onRestart() event");
```

```
            }
            public void onResume()
            {
                super.onResume();
                Log.d(tag, "In the onResume() event");
            }
            public void onPause()
            {
                super.onPause();
                Log.d(tag, "In the onPause() event");
            }
            public void onStop()
            {
                super.onStop();
                Log.d(tag, "In the onStop() event");
            }
            public void onDestroy()
            {
                super.onDestroy();
                Log.d(tag, "In the onDestroy() event");
            }
        }
```

(3) 按F11键在Android模拟器上调试应用程序。

(4) 当活动第一次被加载时，应该可以在LogCat窗口中看到以下内容(单击Debug 透视图，参见图2-3)：

```
12-28 13:45:28.115: DEBUG/Events(334): In the onCreate() event
12-28 13:45:28.115: DEBUG/Events(334): In the onStart() event
12-28 13:45:28.115: DEBUG/Events(334): In the onResume() event
```

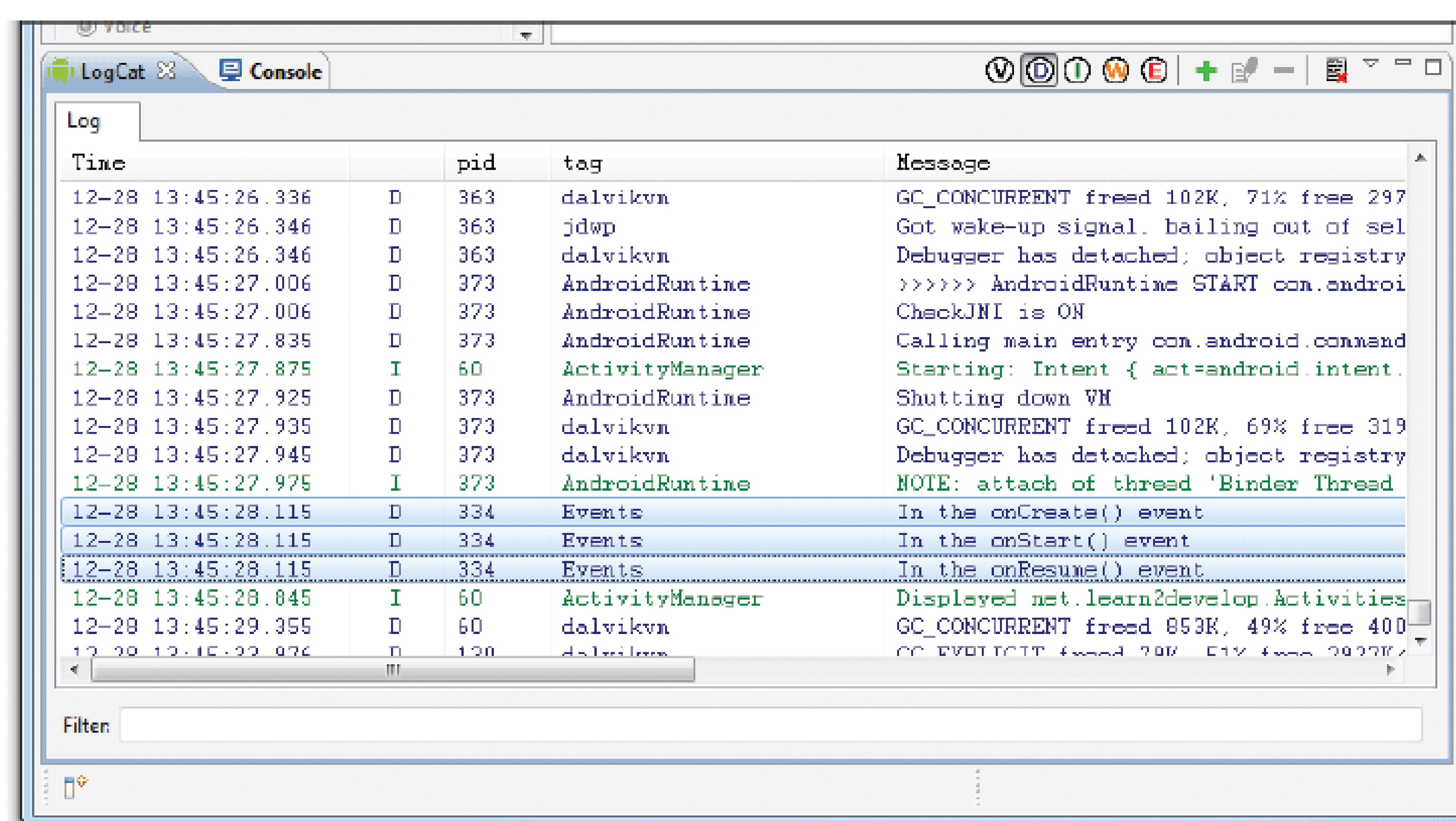


图 2-3

(5) 如果在Android模拟器上按Back按钮，可以观察到以下显示内容：

```
12-28 13:59:46.266: DEBUG/Events(334): In the onPause() event
12-28 13:59:46.806: DEBUG/Events(334): In the onStop() event
```

```
12-28 13:59:46.806: DEBUG/Events(334): In the onDestroy() event
```

(6) 按住Home按钮不放，同时单击Activities图标可以看到以下内容：

```
12-28 14:00:54.115: DEBUG/Events(334): In the onCreate() event
12-28 14:00:54.156: DEBUG/Events(334): In the onStart() event
12-28 14:00:54.156: DEBUG/Events(334): In the onResume() event
```

(7) 按下Android模拟器上的Phone按钮，当前活动就会被推到后台，观察LogCat窗口中的输出：

```
12-28 14:01:16.515: DEBUG/Events(334): In the onPause() event
12-28 14:01:17.135: DEBUG/Events(334): In the onStop() event
```

(8) 注意，onDestroy()事件并没有被调用，表明这个活动仍旧在内存中。按Back按钮退出电话拨号程序，活动又再次显示了。观察LogCat窗口中的输出：

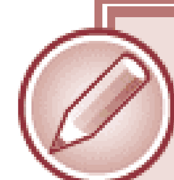
```
12-28 14:02:17.255: DEBUG/Events(334): In the onRestart() event
12-28 14:02:17.255: DEBUG/Events(334): In the onStart() event
12-28 14:02:17.255: DEBUG/Events(334): In the onResume() event
```

onRestart()事件被激活，随后是onStart()和onResume()事件。

示例说明

从这个简单的示例可以看出，当按下Back按钮时，一个活动就被销毁了。知道这一点是至关重要的，因为无论活动当前处于什么状态，它都将丢失。因此，需要在您的活动中额外写一些代码以便在活动要销毁时可以保持其状态(第3章将告诉您怎么做)。在这里，注意onPause()事件在两个情况下都将被调用——当活动被送入后台以及用户按了Back按钮而终止活动时。

当一个活动开始时，onStart()和onResume()事件总是会被调用，而不管这个活动是从后台恢复的还是新创建的。



注意：即使一个应用程序只有一个活动并且这个活动被终止了，该应用程序仍旧会运行于内存中。

2.1.1　如何对活动应用样式和主题

默认情况下，一个活动占据整个屏幕。然而，也可以对活动应用一个对话框主题，使其显示为一个浮动对话框。例如，您打算定制一个活动，以弹出窗口的形式显示它，用来提醒用户将执行的一些操作。在这种情况下，以对话框形式显示活动以引起用户的注意是个不错的方法。

要对活动应用对话框主题，只要修改AndroidManifest.xml文件中的<Activity>元素，添加android:theme属性：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Activities"
      android:versionCode="1"
      android:versionName="1.0">
```

```
        <application android:icon="@drawable/icon"
            android:label="@string/app_name">
            <activity android:name=".MainActivity"
                      android:label="@string/app_name"
                      android:theme="@android:style/Theme.Dialog" >
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category
                        android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            </activity>
        </application>
        <uses-sdk android:minSdkVersion="9" />
    </manifest>
```

这样就可以使活动显示为一个对话框，如图2-4所示。



图 2-4

2.1.2 隐藏活动标题

如果需要的话，还可以隐藏一个活动的标题(例如当您打算向用户显示状态更新时)。要做到这一点，可以使用requestWindowFeature()方法，传递Window.FEATURE_NO_TITLE常量，如下所示：

```
package net.learn2develop.Activities;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

import android.view.Window;
```

```
public class MainActivity extends Activity {
    String tag = "Events";

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //---隐藏标题栏---
        requestWindowFeature(Window.FEATURE_NO_TITLE);

        setContentView(R.layout.main);
        Log.d(tag, "In the onCreate() event");
    }
}
```

这样，标题栏就被隐藏了，如图2-5所示。



图 2-5

2.1.3　显示对话框窗口

您经常会需要显示一个对话框窗口，以便从用户那里得到确认。这时，可以重写在Activity基类中定义的受保护的onCreateDialog()方法来显示一个对话框。下面的“试一试”会告诉您怎么做。

试一试　使用活动显示一个对话框

Dialog.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse创建一个新的Android项目，并将其命名为Dialog。

(2) 将下列粗体显示的语句添加到main.xml文件中：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
<Button
    android:id="@+id/btn_dialog"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Click to display a dialog" />
</LinearLayout>
```

(3) 将下列粗体显示的语句添加到MainActivity.java文件中：

```
package net.learn2develop.Dialog;

import android.app.Activity;
import android.os.Bundle;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {
    CharSequence[] items = { "Google", "Apple", "Microsoft" };
    boolean[] itemsChecked = new boolean [items.length];

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button btn = (Button) findViewById(R.id.btn_dialog);
        btn.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                showDialog(0);
            }
        });
    }
```

```
        @Override
        protected Dialog onCreateDialog(int id) {
            switch (id) {
            case 0:
                return new AlertDialog.Builder(this)
                .setIcon(R.drawable.icon)
                .setTitle("This is a dialog with some simple text...")
                .setPositiveButton("OK", new
                    DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                    int whichButton)
                    {
                        Toast.makeText(getBaseContext(),
                             "OK clicked!", Toast.LENGTH_SHORT).show();
                    }
                })
                .setNegativeButton("Cancel", new
                    DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                        int whichButton)
                    {
                        Toast.makeText(getBaseContext(),
                                "Cancel clicked!", Toast.LENGTH_SHORT).show();
                    }
                })
                .setMultiChoiceItems(items, itemsChecked, new
                    DialogInterface.OnMultiChoiceClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which,
                        boolean isChecked) {
                            Toast.makeText(getBaseContext(),
                                items[which] + (isChecked ? " checked!":
                                " unchecked!"),
                                Toast.LENGTH_SHORT).show();
                        }
                    }
                )
                .create();
            }
            return null;
        }
    }
```

(4) 按F11键在Android模拟器上调试应用程序。单击按钮显示对话框(如图2-6所示)。选中不同的复选框会使Toast类显示那些选中/未选中项的文本。要关闭对话框，可单击OK或Cancel按钮。

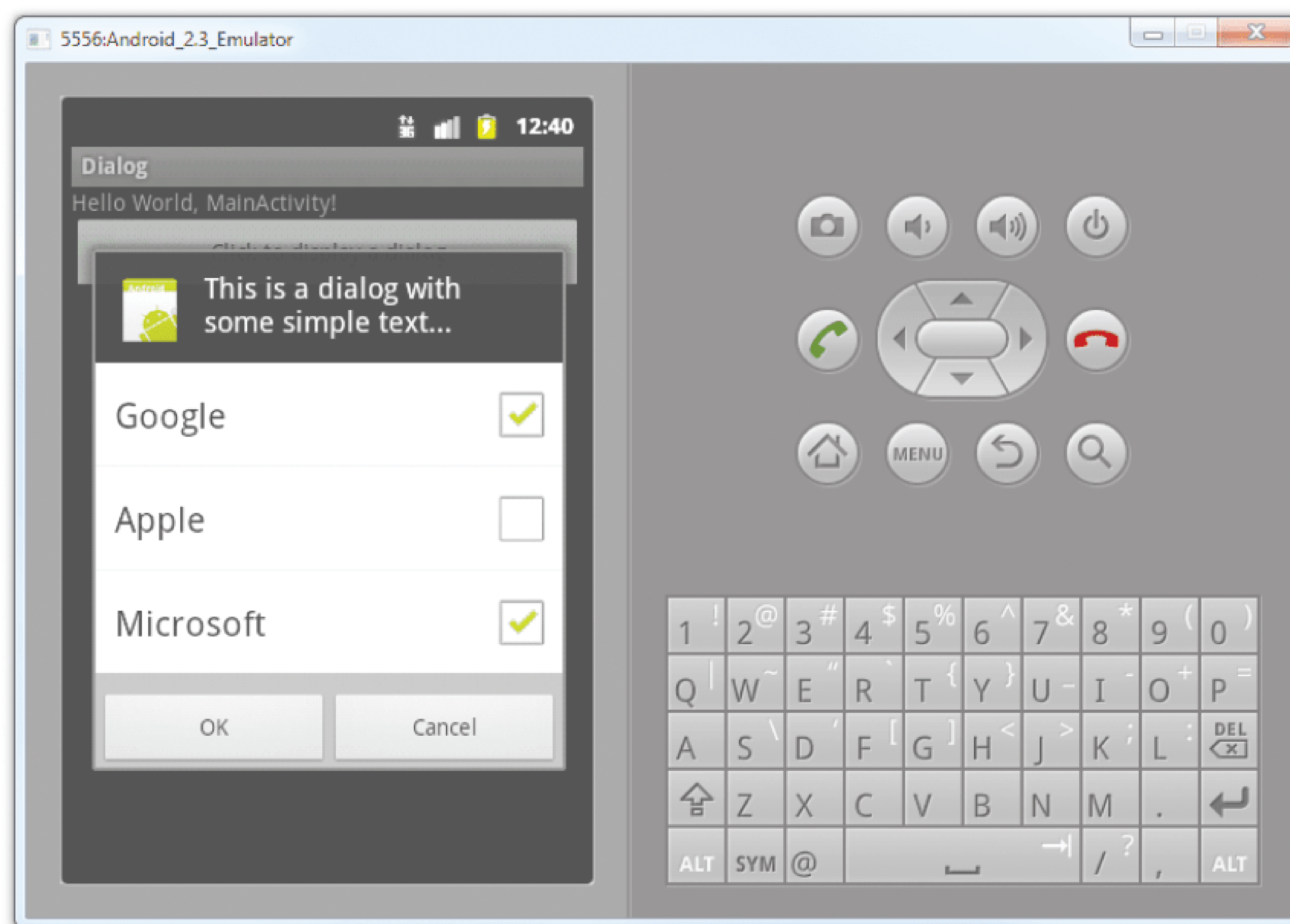


图 2-6

示例说明

要显示一个对话框，首先需要重写Activity类中的onCreateDialog()方法：

```
@Override
protected Dialog onCreateDialog(int id) {
    //...
}
```

在调用showDialog()方法时调用这个方法：

```
Button btn = (Button) findViewById(R.id.btn_dialog);
btn.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        showDialog(0);
    }
});
```

onCreateDialog()是一个用于创建由活动管理的对话框的回调方法。当调用showDialog()方法时，将调用这个方法。showDialog()方法接受一个整型参数，用来标识要显示的特定对话框。

要创建一个对话框，需要使用AlertDialog类的Builder构造函数来设置不同的属性，例如图标、标题、按钮以及复选框：

```
@Override
protected Dialog onCreateDialog(int id) {
    switch (id) {
    case 0:
```

```
                return new AlertDialog.Builder(this)
                .setIcon(R.drawable.icon)
                .setTitle("This is a dialog with some simple text...")
                .setPositiveButton("OK", new
                    DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                    int whichButton)
                    {
                        Toast.makeText(getBaseContext(),
                             "OK clicked!", Toast.LENGTH_SHORT).show();
                    }
                })
                .setNegativeButton("Cancel", new
                    DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                        int whichButton)
                    {
                        Toast.makeText(getBaseContext(),
                                "Cancel clicked!", Toast.LENGTH_SHORT).show();
                    }
                })
                .setMultiChoiceItems(items, itemsChecked, new
                    DialogInterface.OnMultiChoiceClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which,
                        boolean isChecked) {
                            Toast.makeText(getBaseContext(),
                                items[which] + (isChecked ? " checked!":
                                " unchecked!"),
                                Toast.LENGTH_SHORT).show();
                        }
                    }
                )
                .create();
            }
            return null;
        }
```

以上代码使用setPositiveButton()和setNegativeButton()方法分别设置了两个按钮：OK和Cancel。还可以通过setMultiChoiceItems()方法设置一个复选框列表供用户选择。对于setMultiChoiceItems()方法，需要传入两个数组：一个是要显示的项列表，另一个包含了表明每个项是否被选中的值。当选中一个项时，使用Toast类来显示一条信息(如图2-7所示)。

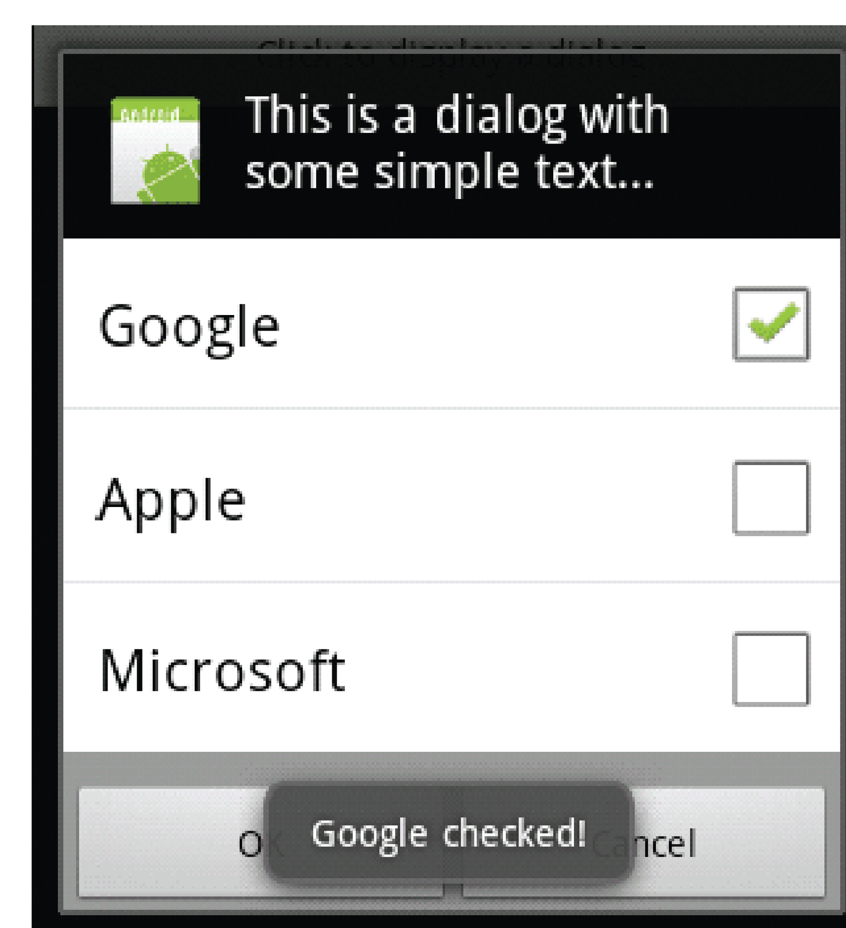


图 2-7

上下文对象

在Android中常常会遇到Context类和其实例。Context类的实例常用来给应用程序提供引用。例如，在本示例中，Toast类的第一个参数接受一个Context对象。

```
        return new AlertDialog.Builder(this)
        .setIcon(R.drawable.icon)
        .setTitle("This is a dialog with some simple text...")
        .setPositiveButton("OK", new
            DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog,
            int whichButton)
            {
                Toast.makeText(getBaseContext(),
                     "OK clicked!", Toast.LENGTH_SHORT).show();
            }
        })
        ...
```

但是，由于Toast()类并没有在活动中直接使用(而是在AlertDialog类中使用)，因此需要使用getBaseContext()方法返回一个Context类的实例。

另一个会遇到Context类的地方是当在一个活动中动态创建视图时。例如，您可能想通过代码动态创建一个TextView视图。因此，使用如下语句实例化TextView类：

```
TextView tv = new TextView(this);
```

TextView类的构造函数接受一个Context对象，因为Activity类是Context类的子类，因此可以使用this关键字来代表这个Context对象。

2.1.4 显示进度对话框

除了前一节介绍的普通对话框外，还可以创建进度对话框。进度对话框对于显示一些活动的进度很有用，如下载操作的状态。

下面的“试一试”将告诉您如何显示一个进度对话框。

试一试　使用活动显示一个进度对话框窗口

(1) 使用前一节创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Dialog;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

```
import android.widget.Toast;

import android.app.ProgressDialog;
import android.os.Handler;
import android.os.Message;

public class MainActivity extends Activity {
    CharSequence[] items = { "Google", "Apple", "Microsoft" };
    boolean[] itemsChecked = new boolean [items.length];

    private ProgressDialog _progressDialog;
    private int _progress = 0;
    private Handler _progressHandler;

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button btn = (Button) findViewById(R.id.btn_dialog);
        btn.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                showDialog(1);
                _progress = 0;
                _progressDialog.setProgress(0);
                _progressHandler.sendEmptyMessage(0);
            }
        });

        _progressHandler = new Handler() {
            public void handleMessage(Message msg) {
                super.handleMessage(msg);
                if (_progress >= 100) {
                    _progressDialog.dismiss();
                } else {
                    _progress++;
                    _progressDialog.incrementProgressBy(1);
                    _progressHandler.sendEmptyMessageDelayed(0, 100);
                }
            }
        };
    }

    @Override
    protected Dialog onCreateDialog(int id) {
        switch (id) {
        case 0:
            return new AlertDialog.Builder(this)
            //...
            //...
            .create();
```

```
            case 1:
                _progressDialog = new ProgressDialog(this);
                _progressDialog.setIcon(R.drawable.icon);
                _progressDialog.setTitle("Downloading files...");
                _progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
                _progressDialog.setButton(DialogInterface.BUTTON_POSITIVE, "Hide", new
                    DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                        int whichButton)
                    {
                        Toast.makeText(getBaseContext(),
                                "Hide clicked!", Toast.LENGTH_SHORT).show();
                    }
                });
                _progressDialog.setButton(DialogInterface.BUTTON_NEGATIVE, "Cancel", new
                    DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                        int whichButton)
                    {
                        Toast.makeText(getBaseContext(),
                                "Cancel clicked!", Toast.LENGTH_SHORT).show();
                    }
                });
                return _progressDialog;
            }
            return null;
        }
    }
```

(2) 按F11键在Android模拟器上对应用程序进行调试。单击按钮显示进度对话框(如图2-8所示)。注意观察进度条将累计到100。

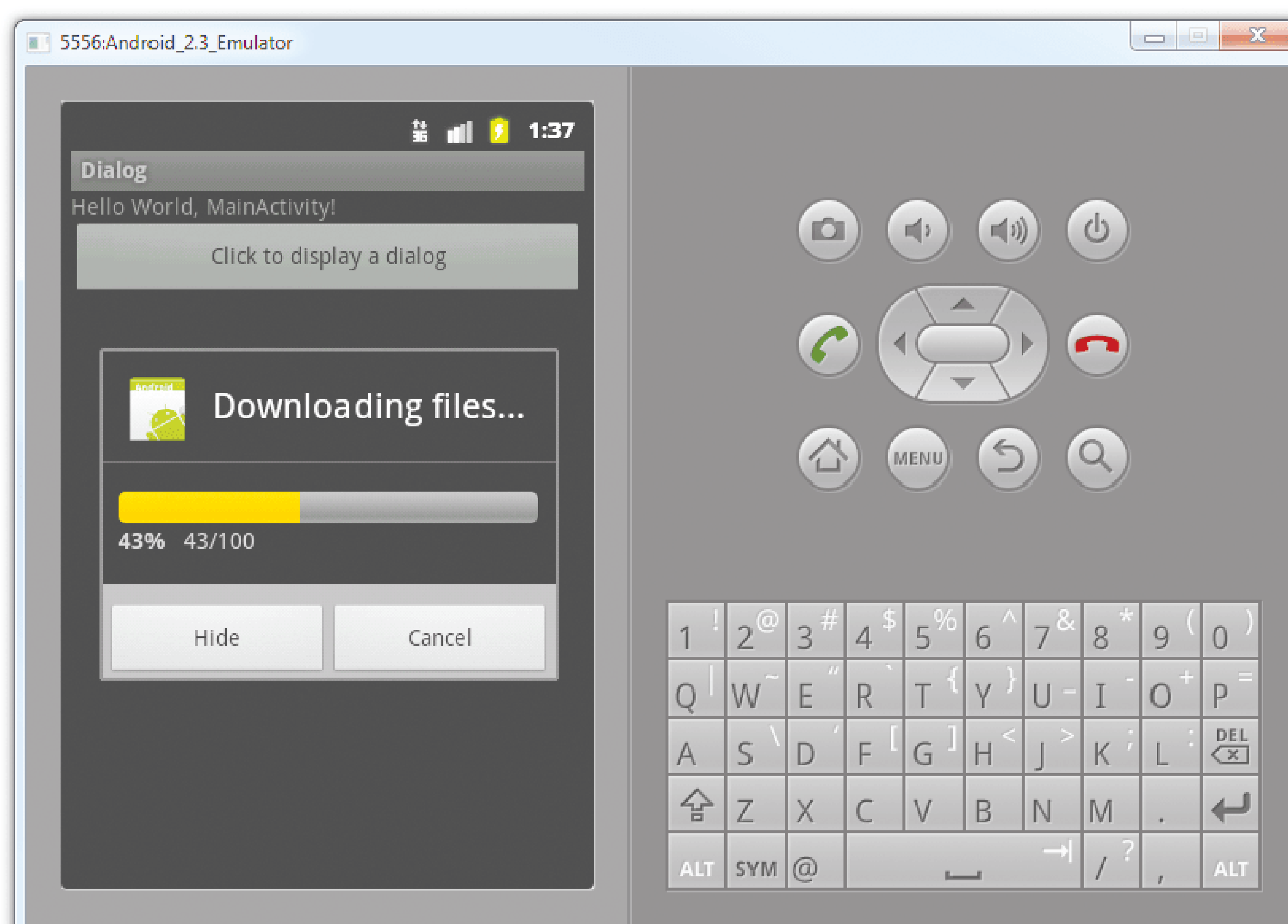


图 2-8

示例说明

为了创建一个进度对话框，首先要创建一个ProgressDialog类的实例并设置其不同的属性，如图标、标题和样式：

```
_progressDialog = new ProgressDialog(this);
_progressDialog.setIcon(R.drawable.icon);
_progressDialog.setTitle("Downloading files...");
_progressDialog.setProgressStyle(ProgressDialog.STYLE_
                HORIZONTAL);
```

然后设置在进度对话框中显示的两个按钮：

```
_progressDialog.setButton(DialogInterface.BUTTON_POSITIVE, "Hide", new
    DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog,
        int whichButton)
    {
        Toast.makeText(getBaseContext(),
                "Hide clicked!", Toast.LENGTH_SHORT).show();
    }
   });
   _progressDialog.setButton(DialogInterface.BUTTON_NEGATIVE, "Cancel", new
       DialogInterface.OnClickListener() {
       public void onClick(DialogInterface dialog,
           int whichButton)
       {
           Toast.makeText(getBaseContext(),
                   "Cancel clicked!", Toast.LENGTH_SHORT).show();
       }
   });
   return _progressDialog;
}
```

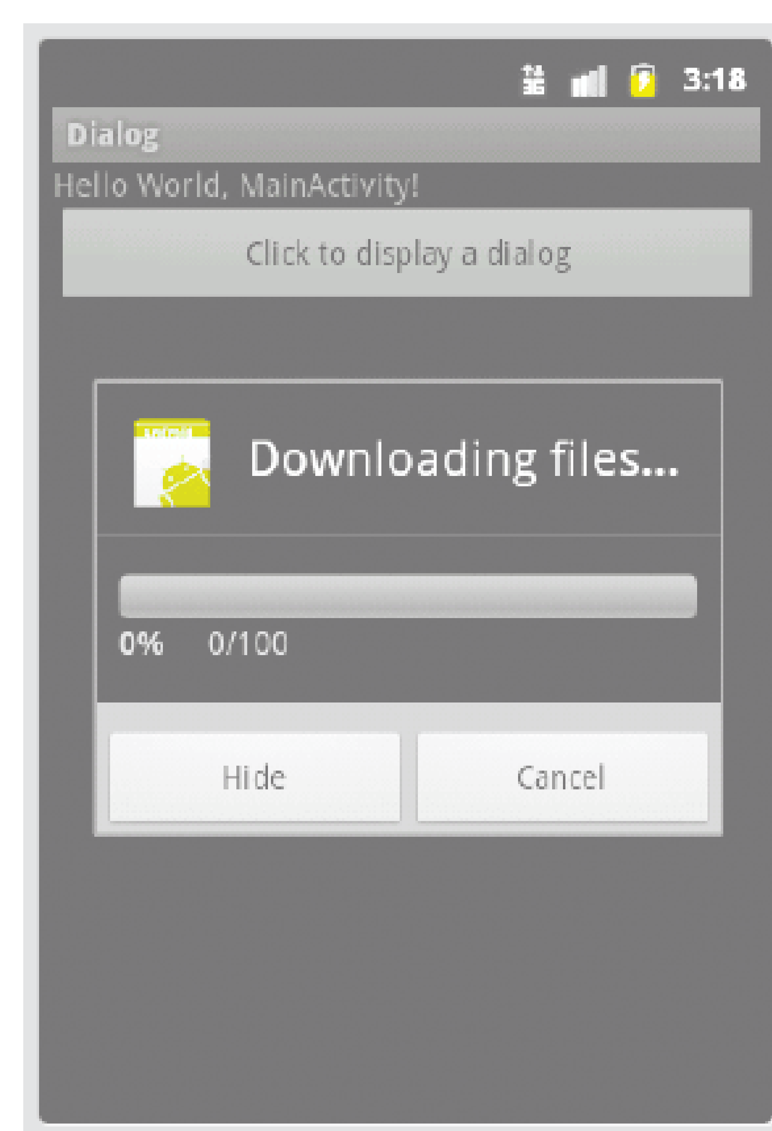


图 2-9

结果就出现了一个进度对话框(如图2-9所示)。

为了在进度对话框中显示进展情况，需要使用一个Handler对象运行一个后台线程：

```
_progress = 0;
_progressDialog.setProgress(0);
_progressHandler.sendEmptyMessage(0)
```

后台线程计数到100，每一次计数延迟100毫秒：

```
_progressHandler = new Handler() {
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        if (_progress >= 100) {
            _progressDialog.dismiss();
        } else {
            _progress++;
```

```
                    _progressDialog.incrementProgressBy(1);
                    _progressHandler.sendEmptyMessageDelayed(0, 100);
                }
            }
        };
```

当计数达到100时，进度对话框将消失。

2.2 使用意图链接活动

一个Android应用程序可以包含零或多个活动。当应用程序具有多个活动时，您可能需要从一个活动导航到另一个活动。在Android中，活动之间的导航通过意图来完成。

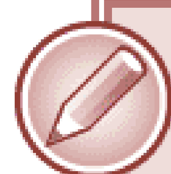
要理解Android中这个非常重要又有些抽象的概念，最好的办法就是亲自去体验一下，看看到底是什么帮您实现了目标。

下面的“试一试”将告诉您如何在一个已有的项目中添加另一个活动，并在这两个活动之间实现导航。

试一试 使用意图链接活动

(1) 使用先前创建的Activities项目，在AndroidManifest.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Activities"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name"
                  android:theme="@android:style/Theme.Dialog" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".Activity2"
                  android:label="Activity 2">
            <intent-filter>
                <action android:name="net.learn2develop.ACTIVITY2" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
```



注意：您需要移除划删除线的属性。

(2) 右击src文件夹下的包名，依次选择New | Class(如图2-10所示)。

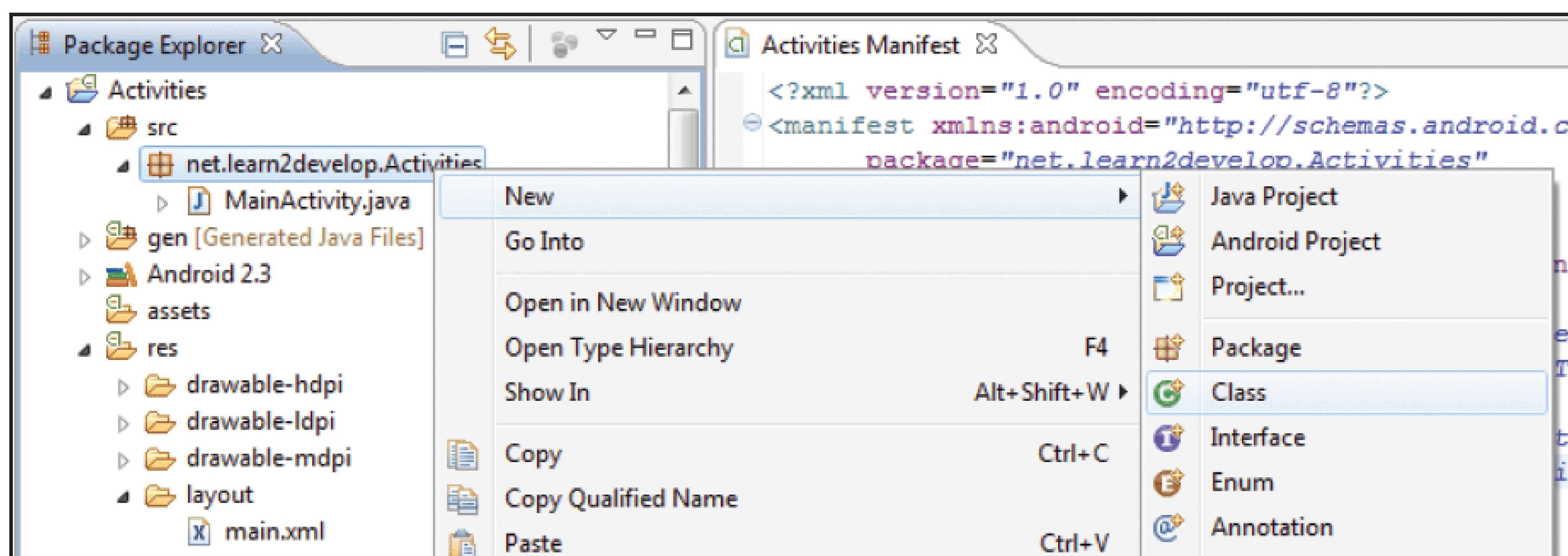


图 2-10

(3) 将新的类文件命名为Activity2(如图2-11所示)，单击Finish按钮。

(4) 右击main.xml文件并选择Copy命令创建一个副本。然后右击res/layout文件夹并选择Paste。将副本文件命名为activity2.xml。现在，res/layout文件夹下就包含了activity2.xml文件(如图2-12所示)。

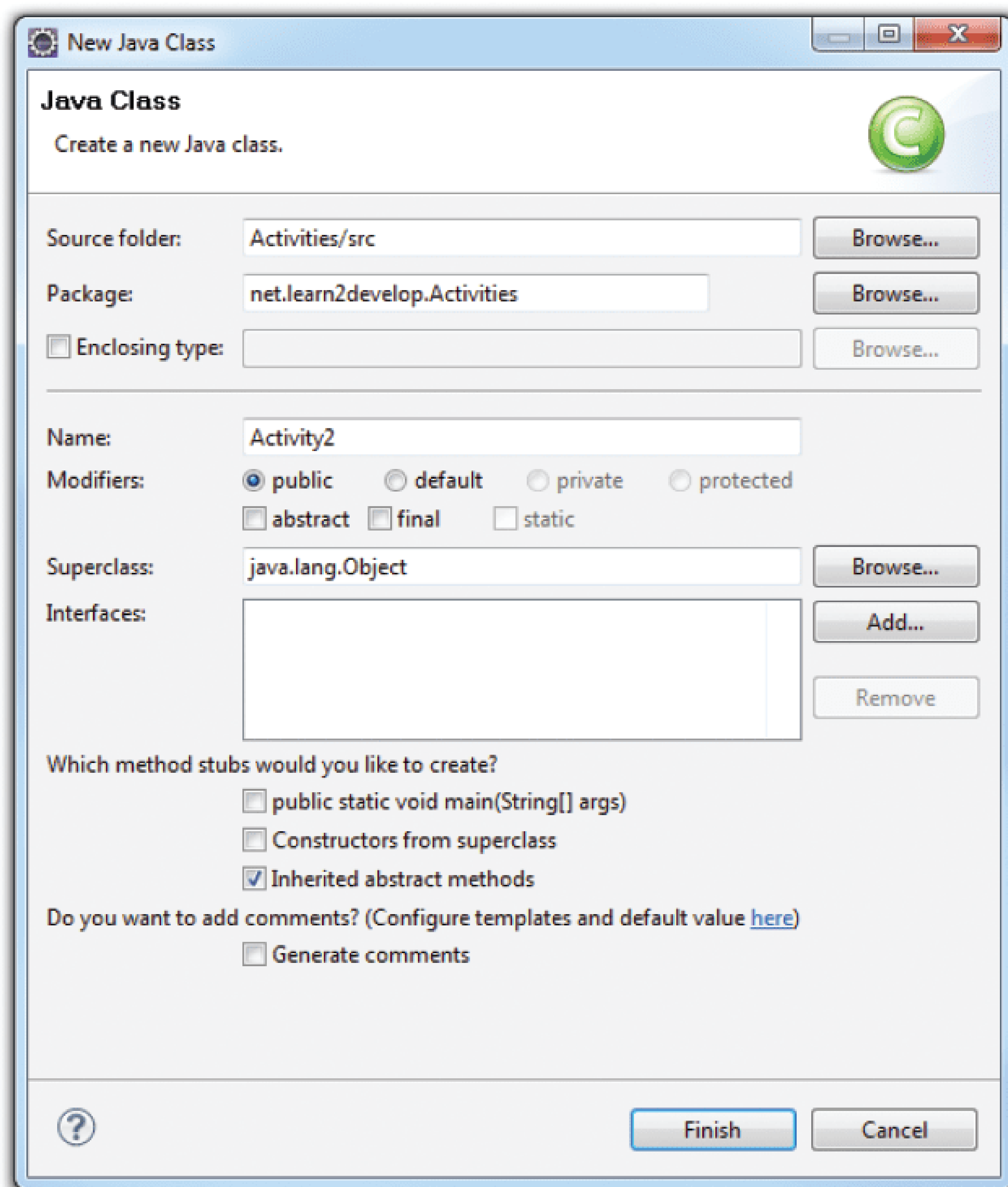


图 2-11

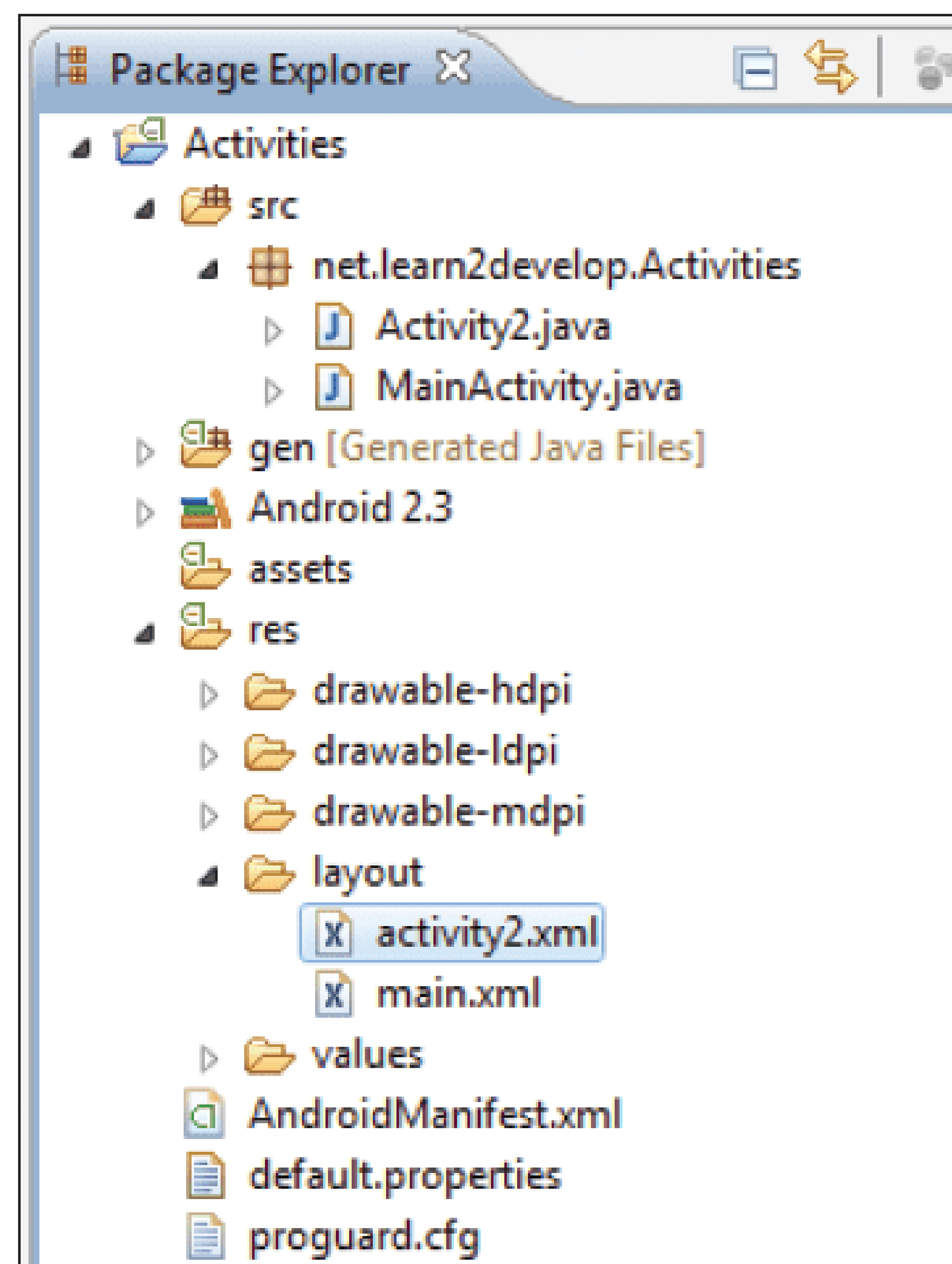


图 2-12

(5) 按如下所示修改activity2.xml文件：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="This is Activity 2!"
        />
    </LinearLayout>
```

(6) 将下列粗体显示的语句添加到Activity2.java文件中：

```
package net.learn2develop.Activities;

import android.app.Activity;
import android.os.Bundle;

public class Activity2 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity2);
    }
}
```

(7) 按下列粗体字内容修改MainActivity.java文件：

```
package net.learn2develop.Activities;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.Window;

import android.view.KeyEvent;
import android.content.Intent;

public class MainActivity extends Activity {
    String tag = "Events";

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //---隐藏标题栏---
        //requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.main);
        Log.d(tag, "In the onCreate() event");
    }
```

```
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER)
        {
            startActivity(new Intent("net.learn2develop.ACTIVITY2"));
        }
        return false;
    }

    public void onStart()     { //... }
    public void onRestart()    { //... }
    public void onResume()    { //... }
    public void onPause()     { //... }
    public void onStop()     { //... }
    public void onDestroy()    { //... }
}
```

(8) 按F11键在Android模拟器上调试应用程序。当第一个活动被加载时，单击方向键盘的中间(如图2-13所示；在一个真实设备上，这个操作可以通过按下轨迹球来完成)。这时，第二个活动将被加载。

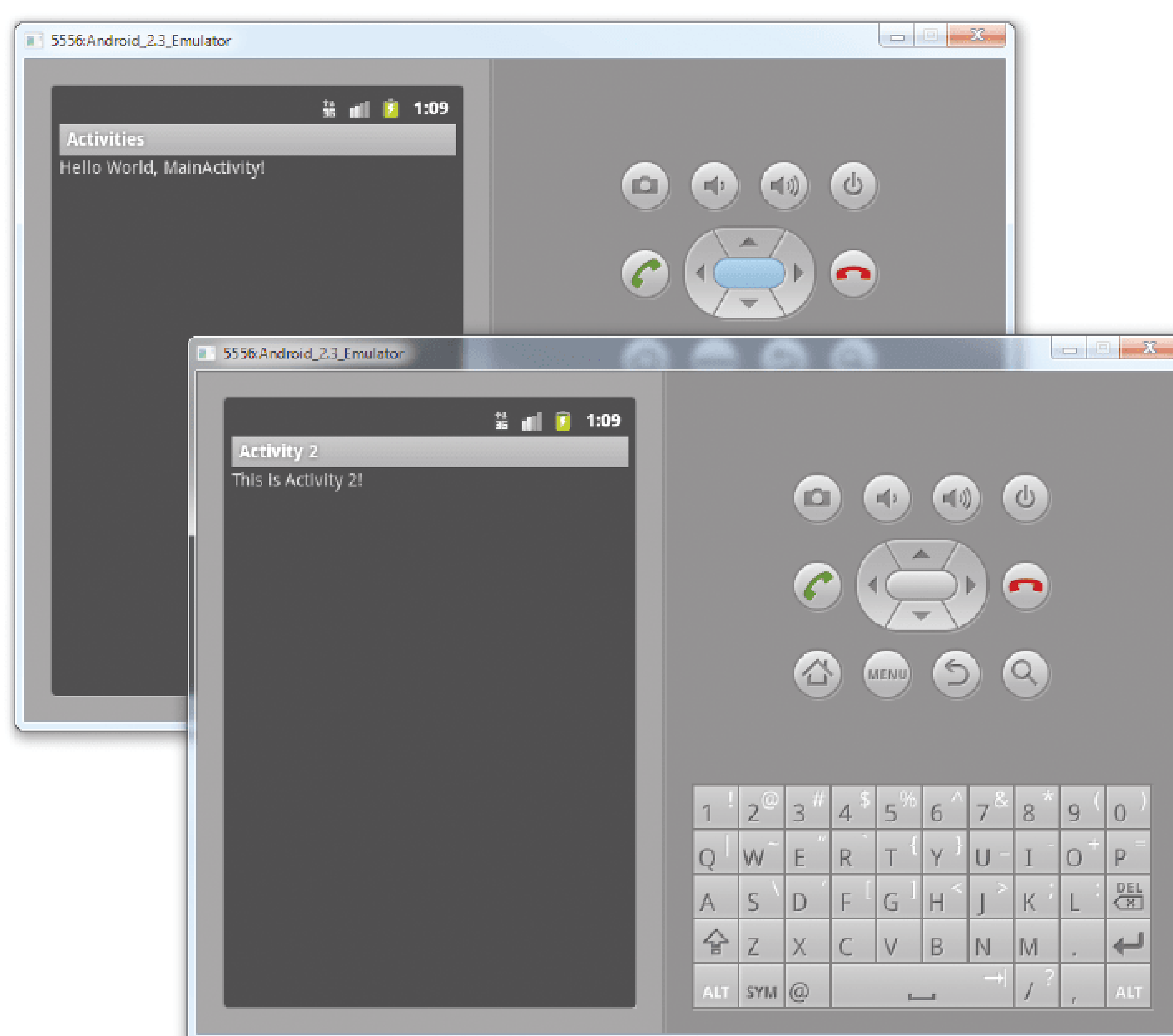


图 2-13

示例说明

正如已经学习过的，一个活动是由一个用户界面组件(例如，main.xml)和一个类组件(例如，MainActivity.java)组成。因此，如果想向项目中添加另外的活动，需要创建这两个组件。

在AndroidManifest.xml文件中，专门添加了以下内容：

```
<activity android:name=".Activity2"
```

```
                android:label="Activity 2">
            <intent-filter>
                <action android:name="net.learn2develop.ACTIVITY2" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
```

到这里，您已经给应用程序添加了一个新的活动。注意以下事项：

- 添加的新活动的名称是Activity2。
- 活动显示的标签名称为Activity2。
- 活动的意图筛选器的名称是net.learn2develop.ACTIVITY2。其他活动将通过这个名称来调用这个活动。理想的情况下，您应该使用您公司的反向域名作为意图筛选器的名称，以减少同另一个应用程序具有相同意图筛选器名称的可能性。
- 意图筛选器的类别是android.intent.category.DEFAULT。您需要将类别添加给意图筛选器，使其他活动可以通过使用startActivity()方法启动这个活动(稍候将作进一步介绍)。

MainActivity.java文件中实现了onKeyDown事件处理程序。无论何时用户按下设备上的一个按键，这个事件都会被触发。当用户按下方向键盘的中间按键时(由KeyEvent.KEYCODE_DPAD_CENTER常量来表示)，您要使用startActivity()方法来显示Activity2，可以通过创建一个Intent类的实例并将Activity2的意图筛选器名称(即net.learn2develop.ACTIVITY2)传递给这个实例来完成：

```
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER)
        {
            startActivity(new Intent("net.learn2develop.ACTIVITY2"));
        }
        return false;
    }
```

Android中的活动可以被设备上运行的任意应用程序调用。例如，可以创建一个新的Android项目，然后使用Activity2的意图筛选器net.learn2develop.ACTIVITY2来显示Activity2。使一个应用程序容易地调用其他应用程序是Android中的基本概念之一。

如果要调用的活动是定义在同一个项目之中的，可以像下面这样重写先前的语句：

```
startActivity(new Intent(this, Activity2.class));
```

不过，只有当您要显示的活动是在同一个项目中作为当前活动时，这种方法才是适用的。

2.2.1 解决意图筛选器的冲突

在先前章节中，我们已经学习了用<intent-filter>元素来定义如何使一个活动被另一个活动所调用。其他活动(在相同或一个单独的应用程序中)如果具有相同的筛选器名称会发生什么呢？例如，假设应用程序有另外一个名为Activity3的活动，在AndroidManifest.xml文件中具有以下入口：

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Activities"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name" >
                  <! -- android:theme="@android:style/Theme.Dialog" -- >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".Activity2"
                  android:label="Activity 2">
            <intent-filter>
                <action android:name="net.learn2develop.ACTIVITY2" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity android:name=".Activity3"
                  android:label="Activity 3">
            <intent-filter>
                <action android:name="net.learn2develop.ACTIVITY2" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
</manifest>
```

如果调用带有下列意图的startActivity()方法，Android操作系统会显示一个选择，如图2-14所示：

```
startActivity(new Intent("net.
    learn2develop.ACTIVITY2"));
```

如果您选中了Use by default for this action项来选择一个活动，那么下一回将再一次调用意图net.learn2develop.ACTIVITY2，它将总是启动先前您选择过的活动。

为了清除这个默认值，转到Android中的Settings应用程序，依次选择Applications | Manage applications，选择应用程序名称(如图2-15所示)。当应用程序的详细信息显示出来时，将屏幕滚动到底部并单击Clear defaults按钮。

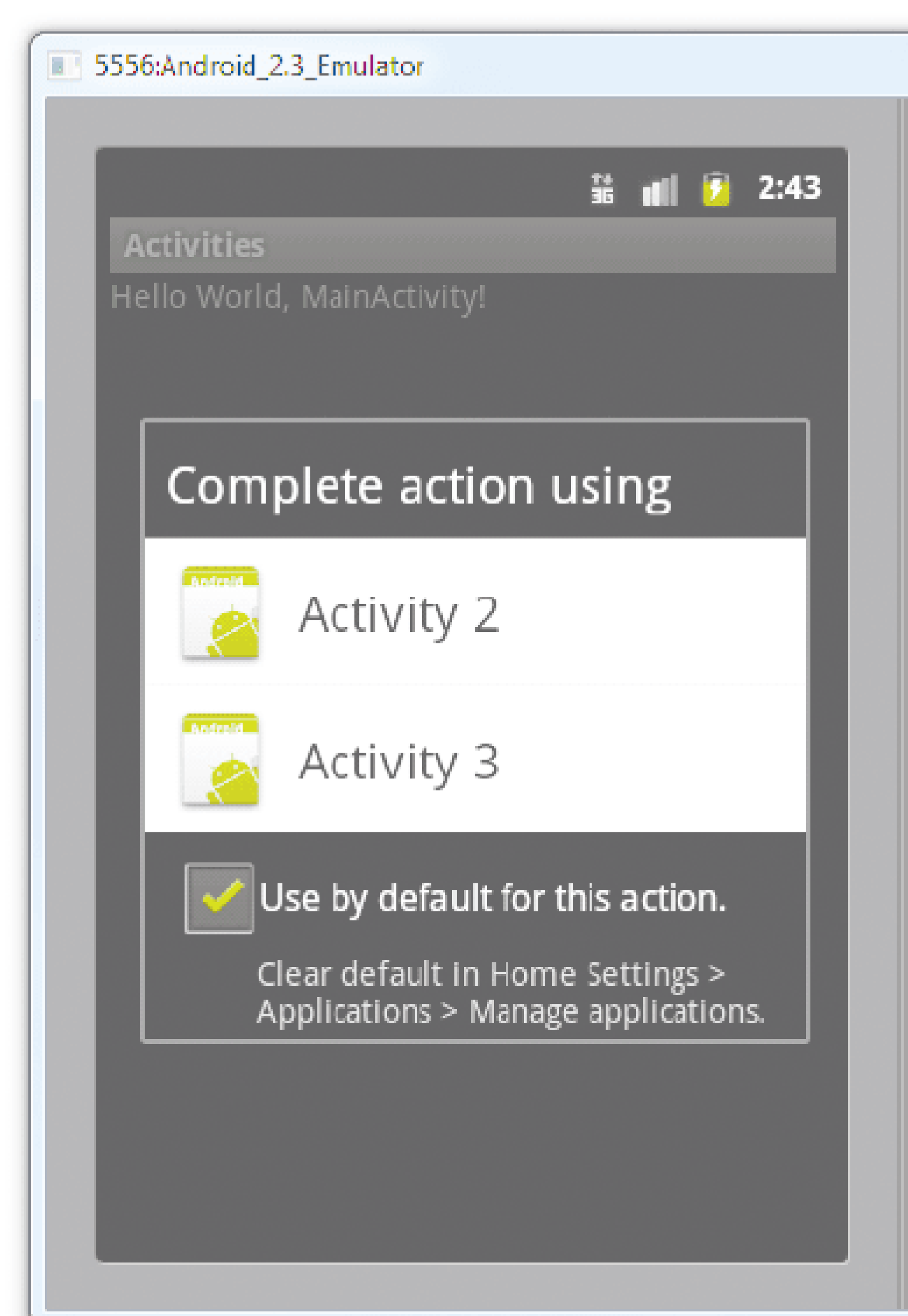


图 2-14

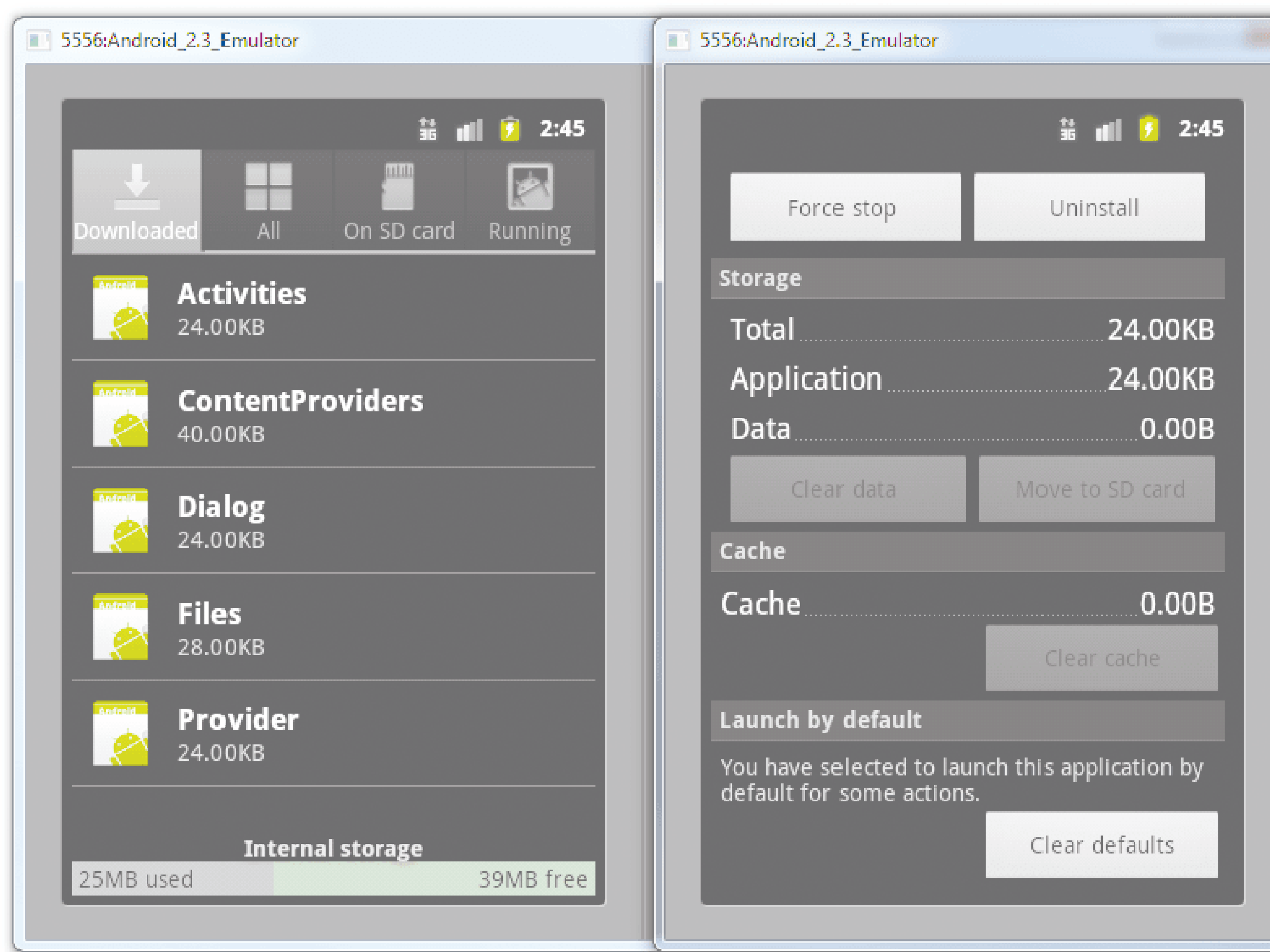


图 2-15

2.2.2　从意图返回结果

startActivity()方法调用另一个活动，但并没有返回结果给当前活动。例如，您可能有一个用于向用户提示用户名和密码的活动。用户在这个活动中输入的信息需要回传给调用它的活动来作进一步的处理。如果需要从一个活动中回传数据，应该使用startActivityForResult()方法。下面的“试一试”演示了这一过程。

试一试　从一个活动获得结果

(1) 使用前一节创建的同一个项目，在main.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Please enter your name" />
<EditText
    android:id="@+id/txt_username"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
<Button
    android:id="@+id/btn_OK"
    android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:text="OK" />
</LinearLayout>
```

(2) 将下列粗体显示的语句添加到Activity2.java文件中：

```
package net.learn2develop.Activities;

import android.app.Activity;
import android.os.Bundle;

import android.content.Intent;
import android.net.Uri;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class Activity2 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity2);

        //---获得OK按钮---
        Button btn = (Button) findViewById(R.id.btn_OK);

        //---OK按钮的事件处理程序---
        btn.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View view) {
                Intent data = new Intent();

                //---获得EditText视图---
                EditText txt_username =
                    (EditText) findViewById(R.id.txt_username);

                //---设置回传的数据---
                data.setData(Uri.parse(
                    txt_username.getText().toString()));
                setResult(RESULT_OK, data);

                //---关闭活动---
                finish();
            }
        });
    }
}
```

(3) 将下列粗体显示的语句添加到MainActivity.java文件中：

```
package net.learn2develop.Activities;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.Window;
import android.view.KeyEvent;
import android.widget.Toast;
import android.content.Intent;

public class MainActivity extends Activity {
    String tag = "Events";
    int request_Code = 1;

    /** 当活动第一次被创建时调用。*/
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //---隐藏标题栏---
        //requestWindowFeature(Window.FEATURE_NO_TITLE);

        setContentView(R.layout.main);
        Log.d(tag, "In the onCreate() event");
    }

    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER)
        {
            //startActivity(new Intent("net.learn2develop.ACTIVITY2"));
            //startActivity(new Intent(this, Activity2.class));

            startActivityForResult(new Intent(
                "net.learn2develop.ACTIVITY2"),
                request_Code);
        }
        return false;
    }

    public void onActivityResult(int requestCode, int resultCode, Intent data)
    {
        if (requestCode == request_Code) {
            if (resultCode == RESULT_OK) {
                Toast.makeText(this,data.getData().toString(),
                    Toast.LENGTH_SHORT).show();
            }
        }
    }

    public void onStart()     { //... }
    public void onRestart()     { //... }
```

```
        public void onResume()    { //... }
        public void onPause()    { //... }
        public void onStop()    { //... }
        public void onDestroy()    { //... }
    }
```

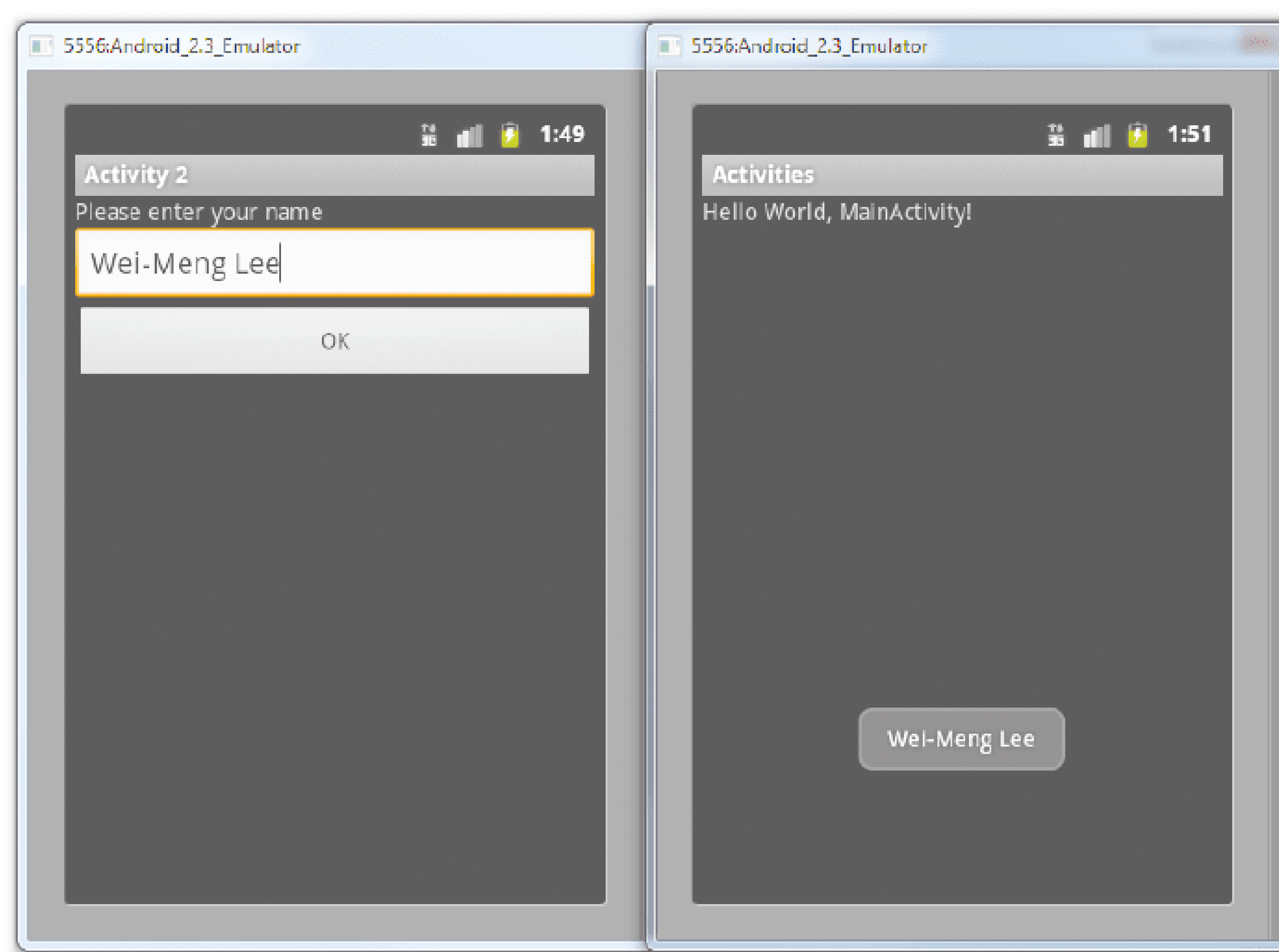


图 2-16

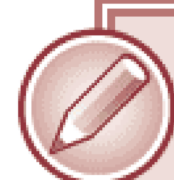
(4) 按F11键在Android模拟器上调试应用程序。当加载第一个活动时，单击方向键盘上中间的按钮，Activity2将被加载。输入您的姓名(如图2-16所示)并单击OK按钮。这时，第一个活动会显示出您用Toast类输入的名字。

示例说明

调用一个活动并等待从此活动返回结果，需要使用startActivityForResult()方法，如下所示：

```
startActivityForResult(new Intent(
                    "net.learn2develop.ACTIVITY2"),
                    request_Code);
```

除了传入一个Intent对象，还需要传入请求码。请求码仅仅是一个整数值，用来标识正在调用的活动。这是必需的，因为当一个活动返回一个值时，必须有办法将它标识出来。例如，您可能同时在调用多个活动，而一些活动可能没有立即返回(如正在等待服务器的应答)。当一个活动返回时，需要这个请求码来确定实际返回的是哪一个活动。



注意：如果请求码设为-1，则使用startActivityForResult()方法来调用活动与使用startActivity()方法来调用是等同的。也就是说，没有结果返回。

为了使被调活动可以返回一个值给调用它的活动，可以通过setData()方法使用一个Intent对象来回传数据：

```
Intent data = new Intent();

//---获得EditText视图---
EditText txt_username =
    (EditText) findViewById(R.id.txt_username);

//---设置回传的数据---
data.setData(Uri.parse(
    txt_username.getText().toString()));
```

```
            setResult(RESULT_OK, data);

            //---关闭活动---
            finish();
```

setResult()方法设置了一个结果码(RESULT_OK或是RESULT_CANCELLED)和回传给调用活动的数据(一个Intent对象)。finish()方法关闭活动并将控制返回给调用者。

在调用者活动中，需要实现onActivityResult()方法，一个活动无论何时返回都要调用这个方法：

```
public void onActivityResult(int requestCode, int resultCode, Intent data)
    {
        if (requestCode == request_Code) {
            if (resultCode == RESULT_OK) {
                Toast.makeText(this,data.getData().toString(),
                    Toast.LENGTH_SHORT).show();
            }
        }
    }
```

这里，检验请求码的正确性，并显示返回的结果。返回的结果通过data参数传入，并且通过getData()方法来获得数据的细节。

2.2.3　使用意图对象传递数据

除了从活动返回数据外，也经常需要传递数据给活动。例如，在前面的示例中，您可能想在活动显示之前在EditText视图中设置一些默认文本。对此，可以使用Intent对象将这些数据传递给目标活动。下面的“试一试”将告诉您如何做：

试一试　将数据传递给目标活动

(1) 使用前面创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
public boolean onKeyDown(int keyCode, KeyEvent event)
{
    if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER)
    {
        //startActivity(new Intent("net.learn2develop.ACTIVITY2"));
        //startActivity(new Intent(this, Activity2.class));
        /*
        startActivityForResult(new Intent(
            "net.learn2develop.ACTIVITY2"),
            request_Code);
        */

        Intent i = new Intent("net.learn2develop.ACTIVITY2");
        Bundle extras = new Bundle();
        extras.putString("Name", "Your name here");
```

```
            i.putExtras(extras);
            startActivityForResult(i, 1);
        }
        return false;
    }
```

(2) 在Activity2.java文件中添加下列粗体显示的语句：

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity2);

        String defaultName="";
        Bundle extras = getIntent().getExtras();
        if (extras!=null)
        {
            defaultName = extras.getString("Name");
        }
        //---获得EditText视图---
        EditText txt_username =
            (EditText) findViewById(R.id.txt_username);
        txt_username.setHint(defaultName);

        //---获得OK按钮---
        Button btn = (Button) findViewById(R.id.btn_OK);

        //---OK按钮的事件处理程序---
        btn.setOnClickListener(new View.OnClickListener()
        {
            //...
        });
    }
```

(3) 按F11键在Android模拟器上调试应用程序。当单击方向键盘的中间按钮时，可注意到目标活动中的EditText视图显示了提示文本(如图2-17所示)。

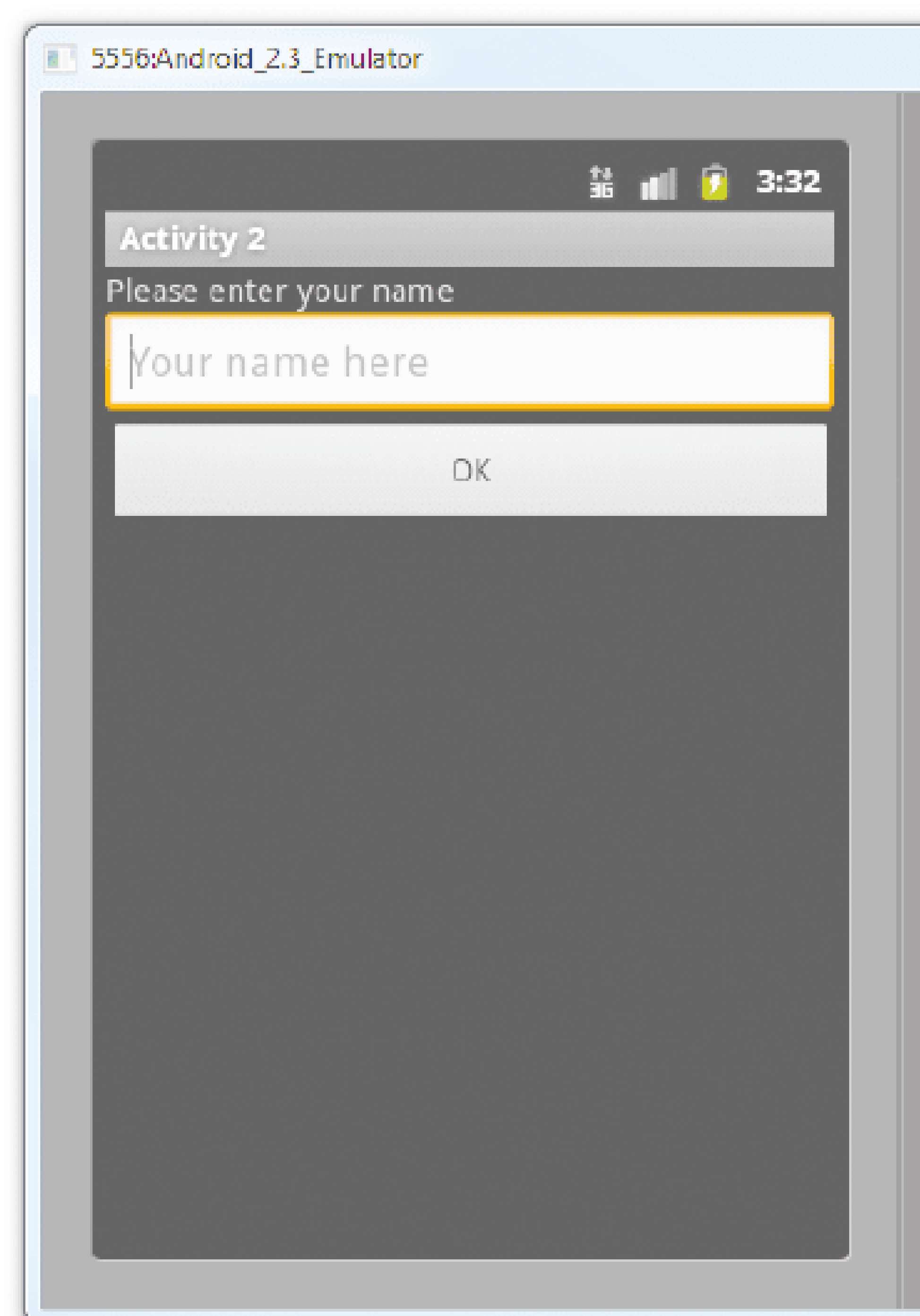
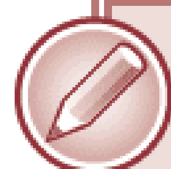


图 2-17



注意：提示文本是EditText视图中常见的占位符文本。它在视图为空时显示，一旦有用户输入，它就会消失。它用于显示提示信息，告诉用户应该输入什么类型的信息。

示例说明

要使用Intent对象给目标活动传递数据，需要使用Bundle对象：

```
            Bundle extras = new Bundle();
```

```
        extras.putString("Name", "Your name here");
        i.putExtras(extras);
```

Bundle对象本质上是一个字典对象，使您可以用键/值对的方式设置数据。在这里，创建一个名为Name的键并赋给其值为Your name here。然后使用putExtras()方法将Bundle对象添加给Intent对象。

在目标活动中，首先使用getIntent()方法来获取启动该活动的意图，然后使用putExtras()方法来获取Bundle对象：

```
Bundle extras = getIntent().getExtras();
if (extras!=null)
{
    defaultName = extras.getString("Name");
}
```

getString()方法从Bundle对象中检索Name键。然后，使用setHint()方法将检索到的字符串赋给EditText视图。

```
//---获得EditText视图---
EditText txt_username =
    (EditText) findViewById(R.id.txt_username);
txt_username.setHint(defaultName);
```

2.3 使用意图调用内置应用程序

到目前为止，您已经了解了如何在自己的应用程序中调用活动。Android编程的关键方面之一就是使用意图从其他应用程序中调用活动。特别是，您的应用程序可以调用Android设备内置的许多应用程序。例如，如果您的应用程序需要使用户能够呼叫保存在Contacts应用程序中特定的一个人，那么您只需要使用一个Intent对象调用Contacts应用程序，用户可以通过它选择要通话的人。这将使您的应用程序具有一致的用户体验，避免再建立另一个应用程序在Contacts应用程序中检索所有的联系人。

下面的“试一试”将演示如何调用Android设备中一些常见的内置应用程序。

试一试　使用意图调用内置应用程序

Intents.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个新的Android项目并命名为Intents。

(2) 在main.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<Button
```

```
        android:id="@+id/btn_webbrowser"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Web Browser" />
    <Button
        android:id="@+id/btn_makecalls"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Make Calls" />
    <Button
        android:id="@+id/btn_showMap"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Show Map" />
    <Button
        android:id="@+id/btn_chooseContact"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Choose Contact" />
    </LinearLayout>
```

(3) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Intents;

import android.app.Activity;
import android.os.Bundle;

import android.content.Intent;
import android.net.Uri;
import android.provider.ContactsContract;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {

    Button b1, b2, b3, b4;
    int request_Code = 1;

    /** 当活动第一次被创建时调用。*/
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //---Web Browser按钮---
        b1 = (Button) findViewById(R.id.btn_webbrowser);
        b1.setOnClickListener(new OnClickListener()
        {
            public void onClick(View arg0){
```

```
                Intent i = new
                    Intent(android.content.Intent.ACTION_VIEW,
                     Uri.parse("http://www.amazon.com"));
                startActivity(i);
            }
        });
        //---Make Calls按钮---
        b2 = (Button) findViewById(R.id.btn_makecalls);
        b2.setOnClickListener(new OnClickListener()
        {
            public void onClick(View arg0){
                Intent i = new
                    Intent(android.content.Intent.ACTION_DIAL,
                  Uri.parse("tel:+651234567"));
                startActivity(i);
            }
        });

        //---Show Map按钮---
        b3 = (Button) findViewById(R.id.btn_showMap);
        b3.setOnClickListener(new OnClickListener()
        {
            public void onClick(View arg0){
                Intent i = new
                    Intent(android.content.Intent.ACTION_VIEW,
                  Uri.parse("geo:37.827500,-122.481670"));
                startActivity(i);
            }
        });

        //---Choose Contact按钮---
        b4 = (Button) findViewById(R.id.btn_chooseContact);
        b4.setOnClickListener(new OnClickListener()
        {
            public void onClick(View arg0){
                Intent i = new
                Intent(android.content.Intent.ACTION_PICK);
                i.setType(ContactsContract.Contacts.CONTENT_TYPE);
                startActivityForResult(i,request_Code);
            }
        });
    }
    public void onActivityResult(int requestCode, int resultCode, Intent data)
    {
        if (requestCode == request_Code)
        {
             if (resultCode == RESULT_OK)
              {
                   Toast.makeText(this,data.getData().toString(),
                       Toast.LENGTH_SHORT).show();
                   Intent i = new Intent(
```

```
                    android.content.Intent.ACTION_VIEW,
                    Uri.parse(data.getData().toString()));
                startActivity(i);
            }
        }
    }
}
```

(4) 按F11键在Android模拟器上调试应用程序。

(5) 单击Web Browser按钮在模拟器中加载Browser应用程序(如图2-18所示)。

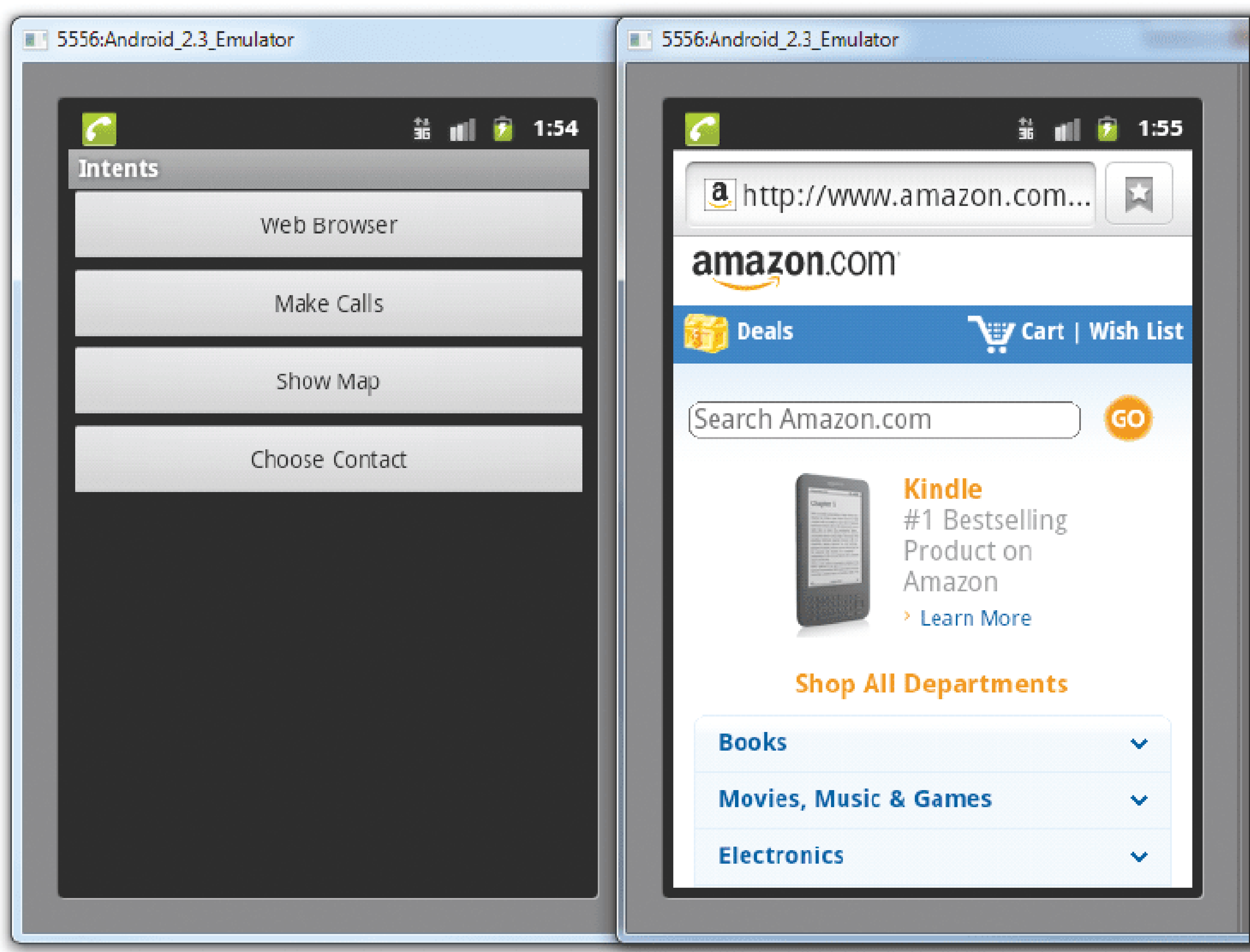


图 2-18

(6) 单击Make Calls按钮，加载Phone应用程序(如图2-19所示)。

(7) 类似地，单击Show Map按钮加载Maps应用程序，如图2-20所示。



图 2-19

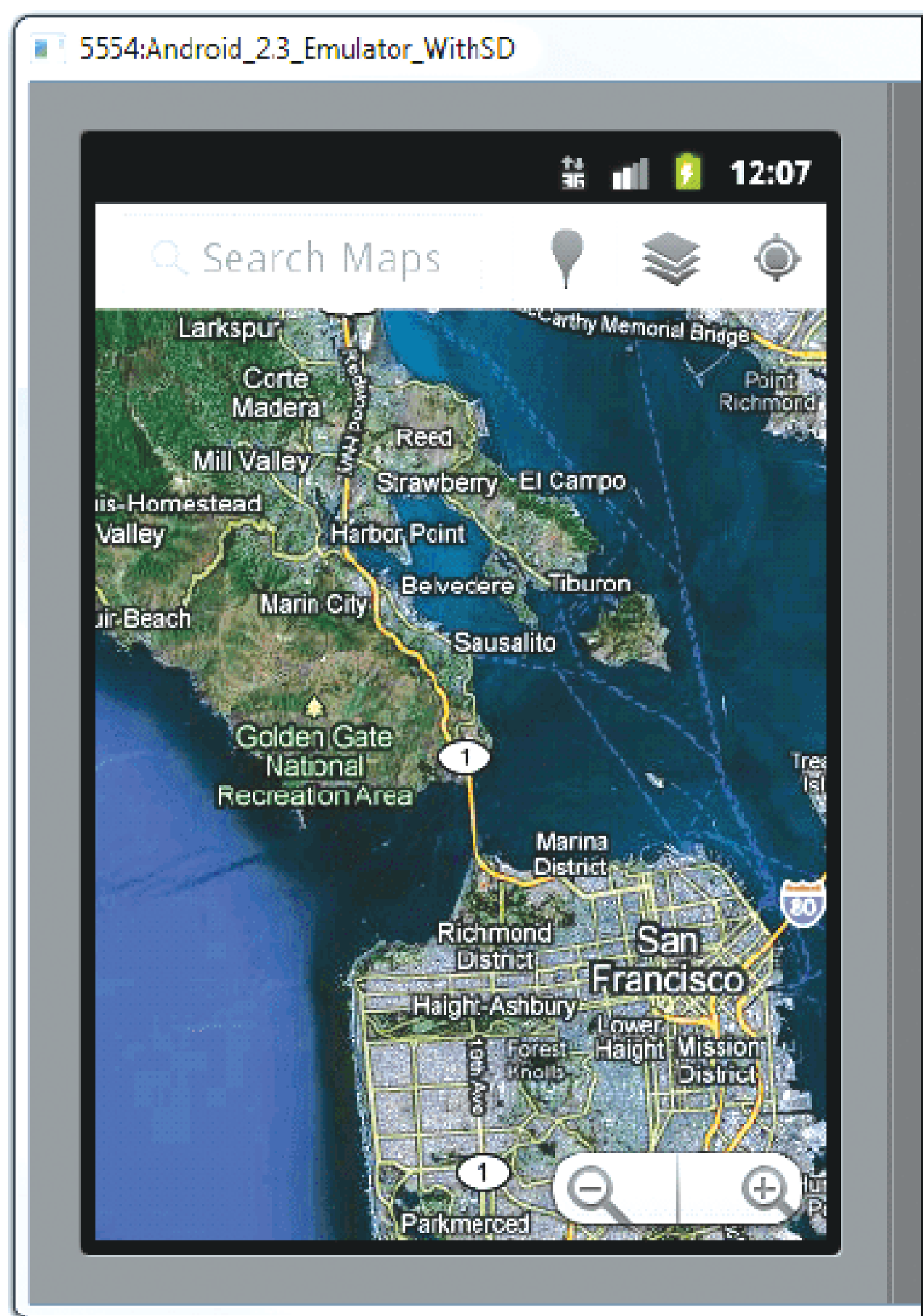
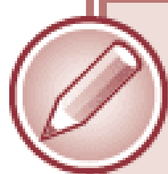


图 2-20



注意：为了显示Maps应用程序，需要在支持Google APIs的AVD上运行它。

(8) 单击Choose Contact按钮显示可选择的联系人列表(如图2-21所示)。选定一个联系人将显示此联系人的详细信息。

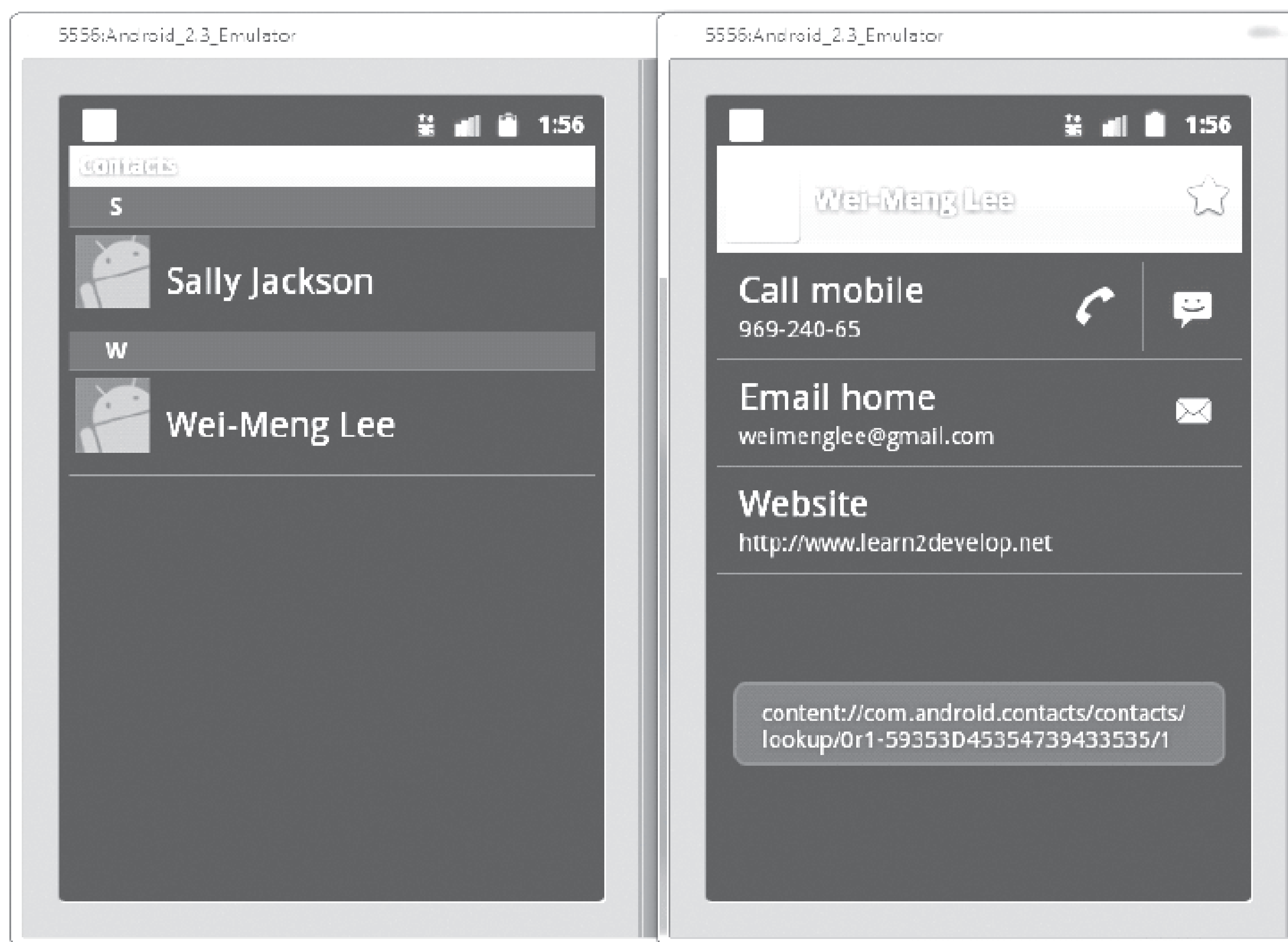


图 2-21

示例说明

本例中，您看到了如何使用Intent类来调用Android中的一些内置应用程序(例如Maps、Phone、Contacts和Browser)。

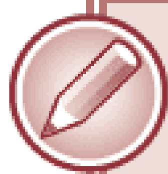
在Android中，意图通常是成对出现的：动作(action)和数据(data)。动作描述了要执行什么，例如编辑一个条目、查看一个条目的内容等。数据则指定了受影响的对象，例如Contacts数据库中的某个人。这一数据被指定为一个Uri对象。

动作的一些示例如下：

- ACTION_VIEW
- ACTION_DIAL
- ACTION_PICK

数据的一些示例如下：

- http://www.google.com
- tel:+651234567
- geo:37.827500,-122.481670
- content://contacts



注意：2.3.2节将介绍可以定义并在活动中使用的数据类型。

动作和数据对共同描述了要执行的操作。例如，要拨一个电话号码，使用ACTION_DIAL/

tel:+651234567对；要显示存储于手机中的联系人列表，使用ACTION_VIEW/content://contacts对；要从联系人列表中选择一个联系人，使用ACTION_PICK/content://contacts对。

在第1个按钮中，创建了一个Intent对象，然后给它的构造函数传递了两个参数——动作和数据：

```
Intent i = new
    Intent(android.content.Intent.ACTION_VIEW,
      Uri.parse("http://www.amazon.com"));
startActivity(i);
```

这里的动作由android.content.Intent.ACTION_VIEW常量表示。使用Uri类的parse()方法将一个URL字符串转换为一个Uri对象。

android.content.Intent.ACTION_VIEW常量实际上指的是android.intent.action.VIEW动作，所以前述代码可以重写为：

```
Intent i = new
    Intent("android.intent.action.VIEW",
        Uri.parse("http://www.amazon.com"));
startActivity(i);
```

前面的代码片段还可以按如下方式改写：

```
Intent i = new
    Intent("android.intent.action.VIEW");
i.setData(Uri.parse("http://www.amazon.com"));
startActivity(i);
```

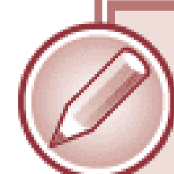
在这里，我们使用setData()方法单独设置了数据。

对于第2个按钮，我们通过在数据部分传入一个电话号码来拨出一个特定的号码：

```
Intent i = new
    Intent(android.content.Intent.ACTION_DIAL,
        Uri.parse("tel:+651234567"));
startActivity(i);
```

这时，拨号程序将显示被呼叫的号码。用户仍旧要按拨号按钮来拨出这个号码。如果想无需用户干预而直接拨出号码，则要修改动作如下：

```
Intent i = new
    Intent(android.content.Intent.ACTION_CALL,
        Uri.parse("tel:+651234567"));
startActivity(i);
```



注意：如果想让应用程序直接呼叫特定号码，需要为应用程序添加android.permission.CALL_PHONE权限。

如果仅仅只是显示拨号程序，而不指定任何号码，只要像下面这样省略数据部分即可：

```
Intent i = new
    Intent(android.content.Intent.ACTION_DIAL);
startActivity(i);
```

第3个按钮使用Action_VIEW常量显示了一个地图：

```
Intent i = new
    Intent(android.content.Intent.ACTION_VIEW,
        Uri.parse("geo:37.827500,-122.481670"));
startActivity(i);
```

这里，要使用geo模式来代替http。

第4个按钮调用Contacts应用程序来使得用户可以选择一个联系人。由于是让用户来选择一个联系人，因此需要Contacts应用程序返回一个值。在这里，需要设置数据的类型以表明需要返回哪种数据：

```
Intent i = new
    Intent(android.content.Intent.ACTION_PICK);
i.setType(ContactsContract.Contacts.CONTENT_TYPE);
startActivityForResult(i,request_Code);
```

如果只是想查看和选择那些有电话号码的联系人，可以按如下方式设置类型：

```
i.setType(
    ContactsContract.CommonDataKinds.Phone.CONTENT_TYPE);
```

这样，将显示联系人和他们的电话号码(如图2-22所示)。



图 2-22

由于希望从Contacts应用程序返回结果，因此使用startActivityForResult()方法，传入Intent对象和请求码来调用它。需要实现onActivityResult()方法来从活动中获得结果：

```
public void onActivityResult(int requestCode,
int resultCode, Intent data)
{
    if (requestCode == request_Code)
    {
         if (resultCode == RESULT_OK)
           {
               Toast.makeText(this,data.getData().toString(),
```

```
                        Toast.LENGTH_SHORT).show();
                    Intent i = new Intent(
                            android.content.Intent.ACTION_VIEW,
                            Uri.parse(data.getData().toString()));
                    startActivity(i);
                }
            }
        }
```

在Contacts应用程序中，当(使用ACTION_PICK常量)选择一个特定的联系人时，将返回一个包含所选联系人的URL，如下所示：

```
content://com.android.contacts/contacts/loopup/0r1-1234567890/1
```

获取这个URL没有多大用处，除非您知道用它来做什么。因此，在这种情况下，可以创建另一个Intent对象来查看它：

```
Intent i = new Intent(
        android.content.Intent.ACTION_VIEW,
        Uri.parse(data.getData().toString()));
startActivity(i);
```

这将显示被选择联系人的详细信息。

2.3.1　理解意图对象

到目前为止，您已经了解了Intent对象调用其他活动的作用。现在正好可以回顾一下并对Intent对象如何施展它的魔力获得更详细的认识。

首先，我们可以通过传递一个动作给一个Intent对象的构造函数来调用另一个活动：

```
startActivity(new Intent("net.learn2develop.ACTIVITY2"));
```

动作(本例中是net.learn2develop.ACTIVITY2)也被称为组件名称，用来标识所要调用的目标活动/应用程序。也可以通过指定存在于项目中的活动的类名修改组件的名称，如下面的代码所示：

```
startActivity(new Intent(this, Activity2.class));
```

还可以通过传入一个动作常量和数据来创建一个Intent对象，例如：

```
Intent i = new
    Intent(android.content.Intent.ACTION_VIEW,
      Uri.parse("http://www.amazon.com"));
startActivity(i);
```

动作部分定义了您要干什么，而数据部分包含了目标活动执行的数据。也可以使用setData()方法传递数据给Intent对象：

```
Intent i = new
    Intent(android.content.Intent.ACTION_VIEW);
i.setData(Uri.parse("http://www.amazon.com"));
```

在本例中，您表示要查看指定URL的Web页面。Android操作系统会查找所有可以满足您要求的活动。这一过程被称为意图解析(intent resolution)。2.3.2节将详细讨论您的活动是如何成为其他活动的目标的。

对于某些意图是无须指定数据的。例如，要从Contacts应用程序中选择一个联系人，可指定该动作，然后使用setType()方法表明MIME类型：

```
Intent i = new
    Intent(android.content.Intent.ACTION_PICK);
i.setType(ContactsContract.Contacts.CONTENT_TYPE);
```

setType()方法显式指定了MIME数据类型来表明要返回的数据类型。ContactsContract.Contacts.CONTENT_TYPE的MIME类型是vnd.android.cursor.dir/contact。

除了指定动作、数据和类型外，Intent对象还可以指定类别。将活动按类别分组为逻辑单元，可以实现Android对活动的进一步筛选。2.3.2节将更详细地讨论类别。

总的来说，Intent对象可以包含以下信息：

- 动作
- 数据
- 类型
- 类别

2.3.2 使用意图筛选器

之前，您已经看到了一个活动是如何使用Intent对象调用另一个活动的。为了使其他活动调用您的活动，需要在AndroidManifest.xml文件的<intent-filter>元素中指定动作和类别，如下所示：

```
<intent-filter>
    <action android:name="net.learn2develop.ACTIVITY2" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

这是一个活动使用net.learn2develop.ACTIVITY2动作调用另一个活动的非常简单的示例。下面的“试一试”则提供了一个更复杂的示例。

试一试 更详细地指定意图筛选器

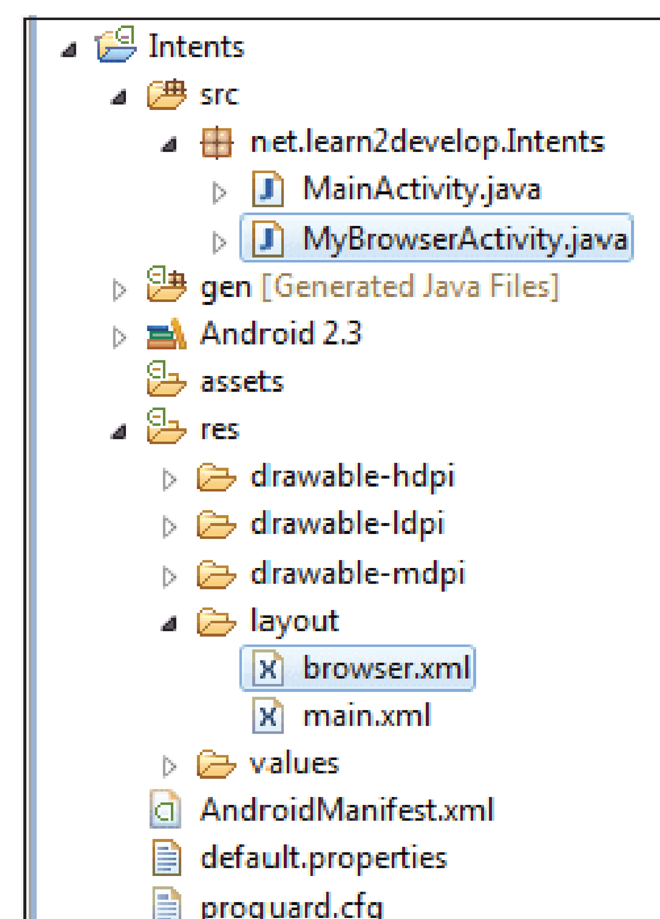


图 2-23

(1) 使用先前创建的Intents项目，给该项目添加一个新类，并命名为MyBrowserActivity.java。在res/layout文件夹下再新增一个XML文件，命名为browser.xml(如图2-23所示)。

(2) 在AndroidManifest.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/
apk/res/android"
      package="net.learn2develop.Intents"
```

```
        android:versionCode="1"
        android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MyBrowserActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <action android:name="net.learn2develop.MyBrowser" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:scheme="http" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

(3) 在main.xml文件中添加下列粗体显示的代码：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<Button
    android:id="@+id/btn_webbrowser"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Web Browser" />
<Button
    android:id="@+id/btn_makecalls"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Make Calls" />
<Button
    android:id="@+id/btn_showMap"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Show Map" />
<Button
    android:id="@+id/btn_chooseContact"
    android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Choose Contact" />
    <Button
        android:id="@+id/btn_launchMyBrowser"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Launch My Browser" />
</LinearLayout>
```

(4) 在MainActivity.java文件中添加下列粗体显示的语句:

```
package net.learn2develop.Intents;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {

    Button b1, b2, b3, b4, b5;
    int request_Code = 1;

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //---Web Browser按钮---
        b1 = (Button) findViewById(R.id.btn_webbrowser);
        b1.setOnClickListener(new OnClickListener()
        {
            //...
        });

        //---Make Calls按钮---
        b2 = (Button) findViewById(R.id.btn_makecalls);
        b2.setOnClickListener(new OnClickListener()
        {
            //...
        });

        //---Show Map按钮---
        b3 = (Button) findViewById(R.id.btn_showMap);
```

```
        b3.setOnClickListener(new OnClickListener()
        {
            //...
        });

        //---Choose Contact按钮---
        b4 = (Button) findViewById(R.id.btn_chooseContact);
        b4.setOnClickListener(new OnClickListener()
        {
            //...
        });

        b5 = (Button) findViewById(R.id.btn_launchMyBrowser);
        b5.setOnClickListener(new OnClickListener()
        {
            public void onClick(View arg0)
            {
            Intent i = new
                Intent("net.learn2develop.MyBrowser");
            i.setData(Uri.parse("http://www.amazon.com"));
            startActivity(i);
            }
        });
    }

    public void onActivityResult(int requestCode, int resultCode, Intent data)
    {
        //...
    }
}
```

(5) 在browser.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<WebView
    android:id="@+id/WebView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</LinearLayout>
```

(6) 在MyBrowserActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Intents;

import android.app.Activity;
import android.net.Uri;
```

```
import android.os.Bundle;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class MyBrowserActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.browser);

        Uri url = getIntent().getData();
        WebView webView = (WebView) findViewById(R.id.WebView01);
        webView.setWebViewClient(new Callback());
        webView.loadUrl(url.toString());
    }

    private class Callback extends WebViewClient {
        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
            return(false);
        }
    }
}
```

(7) 按F11键在Android模拟器上调试应用程序。

(8) 单击Launch my Browser按钮，将会看到显示着Amazon.com的Web页面的一个新活动(如图2-24所示)。

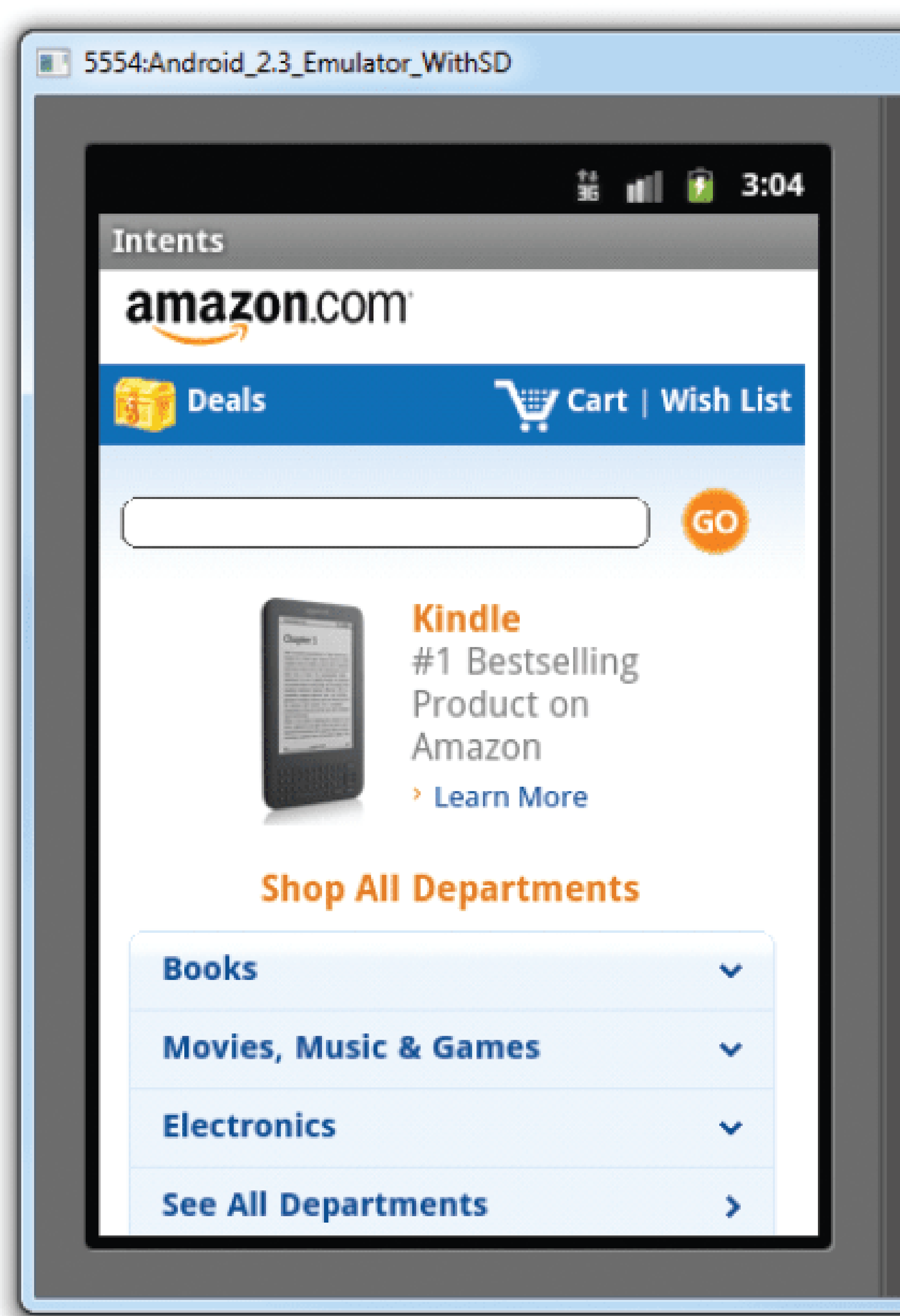


图 2-24

示例说明

在本例中，创建了一个名为MyBrowserActivity的新活动。首先需要在AndroidManifest.xml文件中声明它。

```
<activity android:name=".MyBrowserActivity"
          android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <action android:name="net.learn2develop.MyBrowser" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:scheme="http" />
    </intent-filter>
</activity>
```

在<intent-filter>元素中，声明了活动的两个动作、一个类别以及一个数据。这意味着所有其他活动可以使用android.intent.action.VIEW或net.learn2develop.MyBrowser动作来调用这个活动。对于所有希望别人使用startActivity()或startActivityForResult()方法来调用的活动，它们需要具有android.intent.category.DEFAULT类别。如果没有的话，您的活动将不能被其他活动调用。<data>元素指定了活动期望的数据类型。在本例中，它期望的数据要以http://前缀打头。

先前的意图筛选器还可以按如下方式改写：

```
<activity android:name=".MyBrowserActivity"
          android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:scheme="http" />
    </intent-filter>
    <intent-filter>
        <action android:name="net.learn2develop.MyBrowser" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:scheme="http" />
    </intent-filter>
</activity>
```

按这种方式编写意图筛选器可以对一个意图筛选器中的动作、类别以及数据进行逻辑分组，使其更加具有可读性。

现在如果使用带有如下数据的ACTION_VIEW动作，Android将显示一个选择(如图2-25所示)：

```
Intent i = new
    Intent(android.content.Intent.ACTION_VIEW,
        Uri.parse("http://www.amazon.com"));
startActivity(i);
```

可以在使用Browser应用程序还是当前正在构建的Intents应用程序间进行选择。

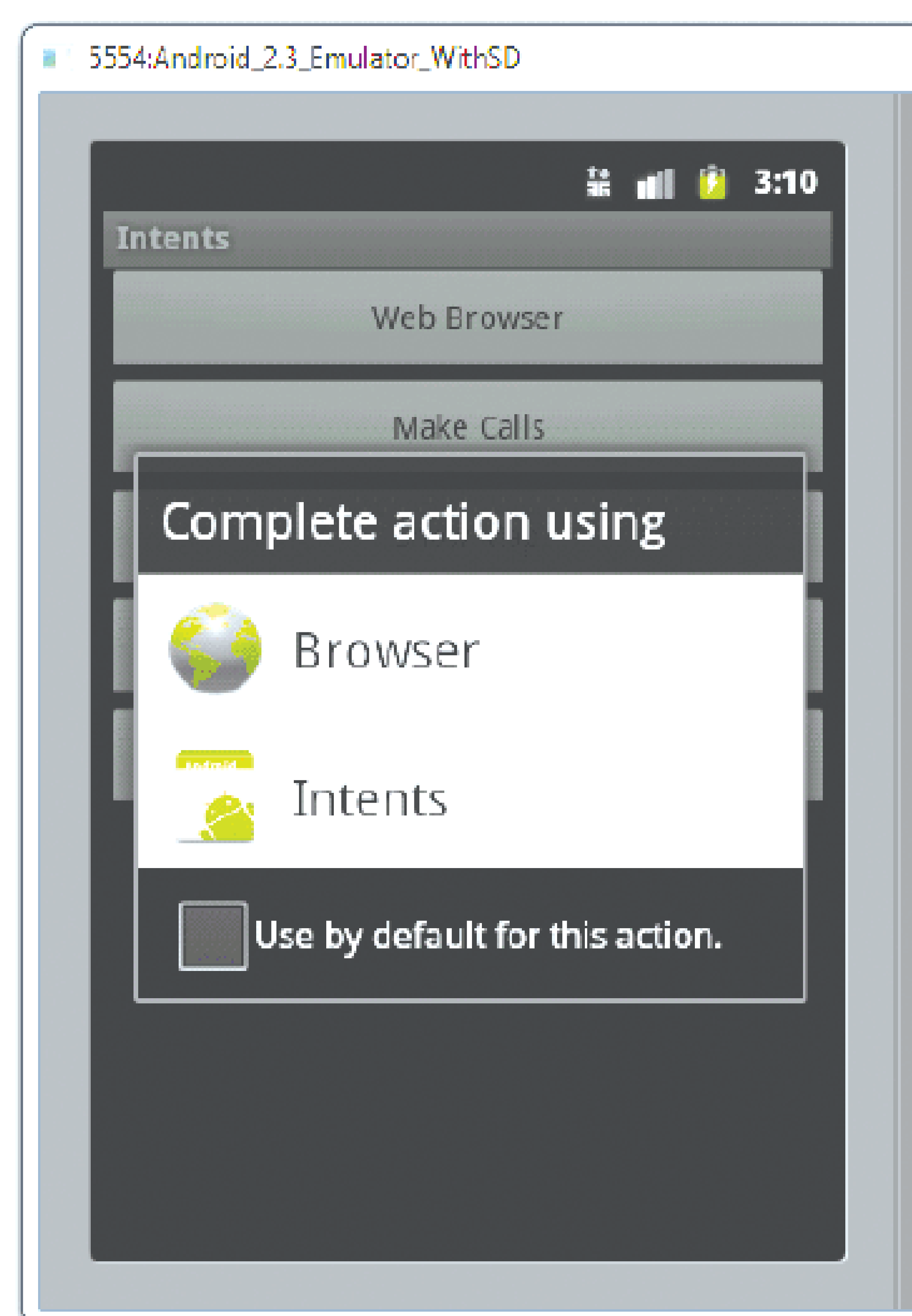


图 2-25

2.3.3 添加类别

可以在意图筛选器中使用<category>元素对活动进行分类。假设已经在AndroidManifest.xml文件中添加了下列<category>元素：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Intents"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MyBrowserActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <action android:name="net.learn2develop.MyBrowser" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="net.learn2develop.Apps" />
                <data android:scheme="http" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

在这种情况下，下列代码将调用MyBrowserActivity活动：

```
Intent i = new
    Intent(android.content.Intent.ACTION_VIEW,
        Uri.parse("http://www.amazon.com"));
i.addCategory("net.learn2develop.Apps");
startActivity(i);
```

使用addCategory()方法将类别添加到Intent对象中。如果遗漏了addCategory()语句，前述的代码仍旧会调用MyBrowserActivity活动，因为它仍旧和默认类别android.intent.category.DEFAULT匹配。

不过，如果指定了一个和意图筛选器中定义的类别不匹配的类别，活动将不会被调用：

```
Intent i = new
    Intent("net.learn2develop.MyBrowser",
        Uri.parse("http://www.amazon.com"));
//---这个类别不匹配意图筛选器中的任何类别---
```

```
i.addCategory("net.learn2develop.OtherApps");
startActivity(i);
```

上述的类别(net.learn2develop.OtherApps)不匹配意图筛选器中的任何类别，所以将引发一个运行时异常。

如果在MyBrowserActivity的意图筛选器中添加前述类别，先前的代码就可以运行了：

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <action android:name="net.learn2develop.MyBrowser" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="net.learn2develop.Apps" />
    <category android:name="net.learn2develop.OtherApps" />
    <data android:scheme="http" />
</intent-filter>
```

可以添加多个类别到一个Intent对象中，以下语句在Intent对象中添加了net.learn2 develop.SomeOtherApps类别：

```
Intent i = new
    Intent("net.learn2develop.MyBrowser",
        Uri.parse("http://www.amazon.com"));
i.addCategory("net.learn2develop.OtherApps");
i.addCategory("net.learn2develop.SomeOtherApps");
startActivity(i);
```

由于意图筛选器没有定义net.learn2develop.SomeOtherApps类别，上述代码将不能调用MyBrowser-Activity活动。为此，需要再一次添加net.learn2develop.SomeOtherApps类别到意图筛选器中。

从这个示例可以看出，在一个活动可以被调用之前，当使用一个带有类别的意图对象时，所有添加到Intent对象的类别必须完全匹配意图筛选器中所定义的类别。

2.4　显示通知

到目前为止，您一直使用Toast类来给用户显示消息。Toast类虽然是一个向用户显示警报的方便的方法，但它不能持久。它只是在屏幕上闪那么几秒钟后就消失掉了。用户如果不一直盯着屏幕，一旦其中包含了重要信息的话，就很容易错过。

对于重要的消息，要使用更加持久化的方法。这种情况下，应当使用NotificationManager在设备顶端的状态栏(有时也称为通知栏)中显示一条持久化的信息。下面的“试一试”演示了如何做到这一点。

试一试　在状态栏上显示通知

Notifications.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个新的Android项目并命名为Notifications。

(2) 在项目的src文件夹下添加一个新的类文件NotificationView.java(如图2-26所示)。另外，在res/layout文件夹下添加一个新的notification.xml文件。

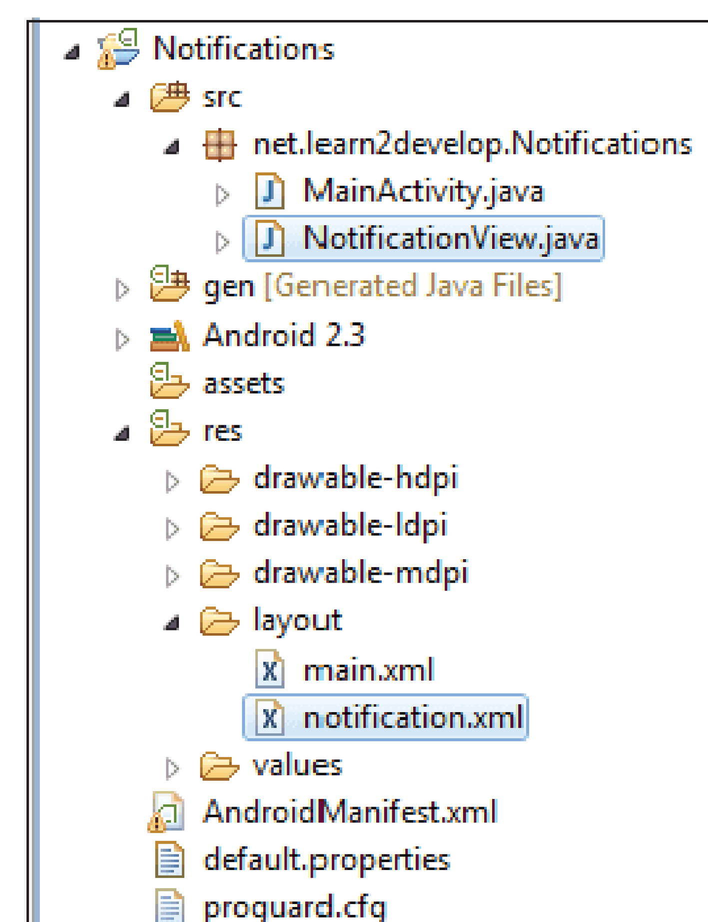


图 2-26

(3) 按如下所示填充notification.xml文件：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.
com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Here are the details for the notification..." />
</LinearLayout>
```

(4) 按如下所示填充NotificationView.java文件：

```
package net.learn2develop.Notifications;

import android.app.Activity;
import android.app.NotificationManager;
import android.os.Bundle;

public class NotificationView extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.notification);

        //---查找通知管理器服务---
        NotificationManager nm = (NotificationManager)
            getSystemService(NOTIFICATION_SERVICE);

        //---取消已经开始的通知
        nm.cancel(getIntent().getExtras().getInt("notificationID"));
    }
}
```

(5) 在AndroidManifest.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Notifications"
      android:versionCode="1"
      android:versionName="1.0">
```

```
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".NotificationView"
            android:label="Details of notification">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
    <uses-permission android:name="android.permission.VIBRATE" />
</manifest>
```

(6) 在main.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<Button
    android:id="@+id/btn_displaynotif"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Display Notification" />
</LinearLayout>
```

(7) 最后，在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Notifications;

import android.app.Activity;
import android.os.Bundle;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {
    int notificationID = 1;
    /** 当活动第一次被创建时调用。 */
```

```
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);

            Button button = (Button) findViewById(R.id.btn_displaynotif);
            button.setOnClickListener(new Button.OnClickListener() {
                public void onClick(View v) {
                    displayNotification();
                }
            });
        }

        protected void displayNotification()
        {
            //---如果用户选择这个通知，使用PendingIntent来启动活动---
            Intent i = new Intent(this, NotificationView.class);
            i.putExtra("notificationID", notificationID);

            PendingIntent pendingIntent =
                PendingIntent.getActivity(this, 0, i, 0);

            NotificationManager nm = (NotificationManager)
                getSystemService(NOTIFICATION_SERVICE);

            Notification notif = new Notification(
                R.drawable.icon,
                "Reminder: Meeting starts in 5 minutes",
                System.currentTimeMillis());

            CharSequence from = "System Alarm";
            CharSequence message = "Meeting with customer at 3pm...";

            notif.setLatestEventInfo(this, from, message, pendingIntent);

            //---延迟100毫秒，震动250毫秒，暂停100毫秒，然后震动500毫秒---
            notif.vibrate = new long[] { 100, 250, 100, 500};
            nm.notify(notificationID, notif);
        }
    }
```

(8) 按F11键在Android模拟器上调试应用程序。

(9) 单击Display Notification按钮(见图2-27左上侧)，状态栏上将出现一个通知。

(10) 单击并向下拖拽状态栏将显示该通知(见图2-27右侧)。

(11) 单击通知将显示NotificationView活动。同时也将把通知从状态栏上清除。

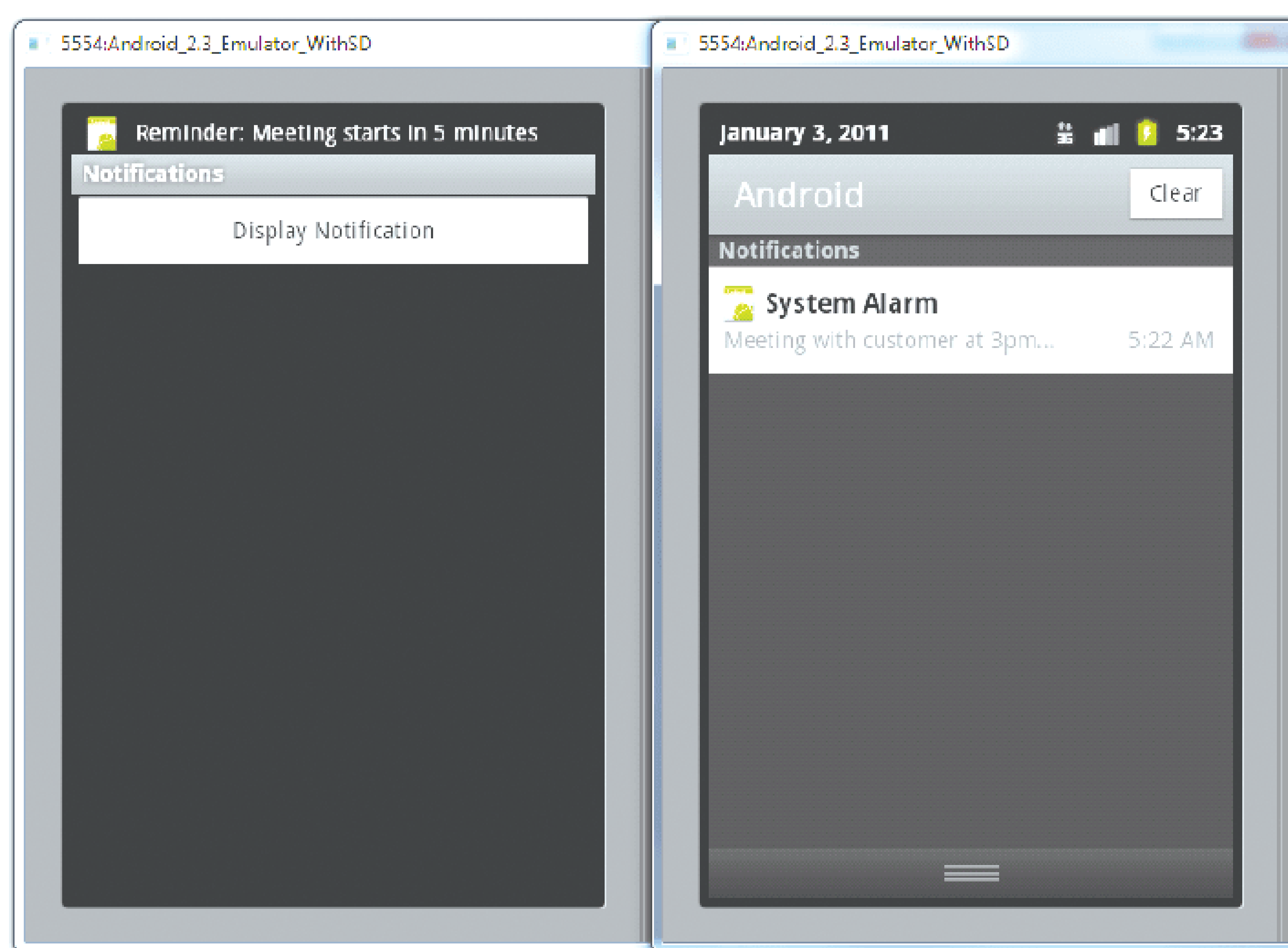


图 2-27

示例说明

要显示一个通知，首先要创建一个指向NotificationView类的Intent对象：

```
//---如果用户选择这个通知，使用PendingIntent来启动活动---
Intent i = new Intent(this, NotificationView.class);
i.putExtra("notificationID", notificationID);
```

当用户从通知列表中选择了一个通知时，这个意图将被用来启动另一个活动。在这个示例中，给Intent对象添加一个键/值对可以用来标记通知ID，标识目标活动的通知。这个ID将在以后用来撤销这个通知。

还需要创建一个PendingIntent对象。PendingIntent对象可以代表应用程序帮助您在后面某个时候执行一个动作，而不用考虑应用程序是否正在运行。在这里，按如下所示初始化它：

```
PendingIntent pendingIntent =
    PendingIntent.getActivity(this, 0, i, 0);
```

getActivity()方法检索一个PendingIntent对象并使用如下参数设置它：

- 上下文——应用程序上下文
- 请求码——用于意图的请求码
- 意图——用来启动目标活动的意图
- 标志——活动启动时使用的标志

然后，获取一个NotificationManger类的实例并创建一个Notification类的实例：

```
NotificationManager nm = (NotificationManager)
    getSystemService(NOTIFICATION_SERVICE);

Notification notif = new Notification(
    R.drawable.icon,
```

```
        "Reminder: Meeting starts in 5 minutes",
        System.currentTimeMillis());
```

当通知首次显示在状态栏上时，Notification类使您能够指定通知的主要信息。Notification构造函数的第二个参数在状态栏上设置了“滚动文本”(如图2-28所示)。

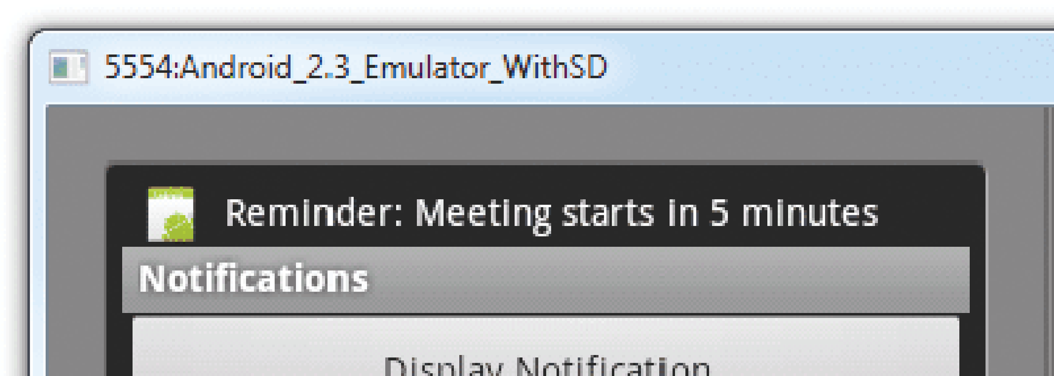


图 2-28

接下来，使用setLatestEventInfo()方法来设置通知的详细内容：

```
CharSequence from = "System Alarm";
CharSequence message = "Meeting with customer at 3pm...";
notif.setLatestEventInfo(this, from, message, pendingIntent);
//---延迟100毫秒，震动250毫秒，暂停100毫秒，然后震动500毫秒---
notif.vibrate = new long[] { 100, 250, 100, 500};
```

上述代码还将通知设置为震动手机。最后，使用notify()方法来显示通知。

```
nm.notify(notificationID, notif);
```

当用户单击通知时，NotificationView活动就会启动。这里，使用NotificationManager对象的cancel()方法并传递给它通知的ID(通过Intent对象传递)来取消这个通知：

```
//---查找通知管理器服务---
NotificationManager nm = (NotificationManager)
    getSystemService(NOTIFICATION_SERVICE);

//---取消已经开始的通知
nm.cancel(getIntent().getExtras().getInt("notificationID"));
```

2.5 本章小结

本章首先详细介绍了活动是如何工作的以及用于显示它们的各种形式。您还了解了如何使用活动来显示对话框窗口。

本章第二部分阐述了Android中的一个非常重要的概念——意图。意图是使不同活动连接起来的“胶水”，也是为Android平台进行开发需要了解的一个关键概念。

练　习

1. 如果有两个或多个活动具有相同的意图筛选器名称，那将会发生什么?
2. 写一段代码来调用内置的Browser应用程序。
3. 在意图筛选器中，可以指定哪些组成部分?
4. Toast类和NotificationManager类的区别是什么?

练习答案参见附录C。

本章主要内容

主　　题	关 键 概 念
创建活动	所有活动必须在AndroidManifest.xml文件中声明
活动的关键生命周期	当活动启动时，总是调用onStart()和onResume()事件。当活动被停止或转入后台时，总是调用onPause()事件
以对话框形式显示活动	使用showDialog()方法并实现onCreateDialog()方法
意图	连接不同活动的“胶水”
意图筛选器	可以使您指定应当如何调用活动的“筛选器”
调用活动	使用startActivity()或startActivityForResult()方法
传递数据给一个活动	使用Bundle对象
Intent对象中的组成部分	Intent对象包含动作、数据、类型和类别
显示通知	使用NotificationManager类
PendingIntent对象	PendingIntent对象可以代表应用程序帮助您在后面某个时候执行一个动作，而不用考虑应用程序是否正在运行

第3章 Android用户界面

本章将介绍以下内容

- 可以用来布置视图的多种视图组(ViewGroup)
- 如何适应屏幕方向的变化
- 如何管理屏幕方向的变化
- 如何以编程方式创建用户界面
- 如何侦听用户界面通知

在第2章中，您已经学习了Activity类和它的生命周期，了解了活动就是用户用来和应用程序交互的一个手段。然而，活动本身并没有在屏幕上呈现。相反，它需要使用视图和视图组来绘制屏幕。本章将学习有关在Android中创建用户界面的详细内容，以及用户是如何与之交互的。此外，还将学习如何处理Android设备屏幕方向的变化。

3.1 了解屏幕的构成

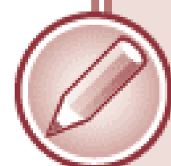
在第2章中，您已经知道了Android应用程序的基本单元是活动。活动显示了应用程序的用户界面，它可以包含按钮、标签、文本框等小部件。通常情况下，使用一个XML文件(例如，位于res/layout文件夹下的main.xml文件)来定义用户界面，类似如下所示：

```
<? xml version="1.0" encoding="utf-8"? >
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

运行时，在Activity类的onCreate()事件处理程序中使用Activity类的setContentView()方法加载XML用户界面：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
}
```

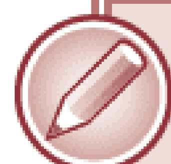
在编译过程中，XML文件中的每一个元素被编译成相应的Android GUI类，这些类使用方法来表示属性。在此文件被加载后，Android系统就会创建活动的用户界面。



注意：尽管使用XML文件构建用户界面通常比较容易，但有时您还需要在运行时动态地构建用户界面(例如，编写游戏时)。因此，完全用代码创建用户界面也是可行的方法。本章稍后将为您提供一个示例，演示是如何实现这一点的。

3.1.1　视图和视图组

活动包含了视图和视图组。视图是一个可以在屏幕上显现的小部件，例如按钮、标签和文本框。视图派生自基类android.view.View。



注意：第4章和第5章将讨论Android中各种常见的视图。

一个或多个视图可以组合成一个视图组。视图组(其本身就是一种特殊的视图类型)提供了一种布局，您可以按该布局定制视图的外观和顺序。视图组的例子包括LinearLayout和FrameLayout。视图组派生于基类android.view.ViewGroup。

Android支持以下视图组：

- LinearLayout
- AbsoluteLayout
- TableLayout
- RelativeLayout
- FrameLayout
- ScrollView

在后面的章节中将对这每一个视图组进行详细讨论。注意，在实践中将不同类型的布局组合起来创建想要的用户界面是很常见的。

3.1.2　LinearLayout

LinearLayout是以单行或单列的形式排列视图的。子视图可以水平或垂直地排列。要了解LinearLayout是如何工作的，请考虑main.xml文件中通常包含的以下元素：

```
<? xml version="1.0" encoding="utf-8"? >
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
```

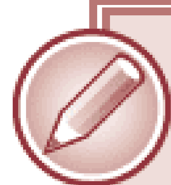
```
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
    </LinearLayout>
```

在main.xml文件中，可注意到根元素<LinearLayout>及其包含的<TextView>元素，<LinearLayout>元素用来控制其所包含的视图的显示顺序。

每个视图和视图组都有一组公共属性，其中一些如表3-1所示。

表3-1　视图和视图组使用的公共属性

属　性	描　述
layout_width	指定视图或视图组的宽度
layout_height	指定视图或视图组的高度
layout_marginTop	指定视图或视图组顶边额外的空间
layout_marginBottom	指定视图或视图组底边额外的空间
layout_marginLeft	指定视图或视图组左边额外的空间
layout_marginRight	指定视图或视图组右边额外的空间
layout_gravity	指定如何定位子视图
layout_weight	指定在布局中应该给视图分配多少额外空间
layout_x	指定视图或视图组的x坐标
layout_y	指定视图或视图组的y坐标



注意：以上某些属性只有当一个视图在特定的视图组中时才适用。例如，layout_weight和layout_gravity属性只适用于视图位于LinearLayout或TableLayout视图组中的情况。

举例来说，使用fill_parent常量可以让<TextView>元素的宽度填满其父元素(本例中指屏幕)的整个宽度。wrap_content常量表明了其高度就是其内容(本例中指其包含的文本)的高度。如果不打算让<TextView>视图占据一整行，可以将其layout_width属性设置为wrap_content，如下所示：

```
    < TextView
        android:layout_width="wrap_content
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
```

这将把视图的宽度设为此视图所包含的文本的宽度。

考虑以下布局：

```
    <? xml version="1.0" encoding="utf-8"? >
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="105dp"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<Button
    android:layout_width="160dp"
    android:layout_height="wrap_content"
    android:text="Button"
    />
</LinearLayout>
```

度量单位

当指定Android用户界面上元素的大小时，应知道以下度量单位：

- dp——与密度无关的像素(density-independent pixel)。160dp相当于1英寸的屏幕物理尺寸。当在布局中指定视图尺寸时，推荐将dp作为度量单位。当指的是与密度无关的像素时，可以使用dp或dip。
- sp——与比例无关的像素(scale-independent pixel)。与dp类似，推荐用于指定字体大小。
- pt——磅。1磅等于1/72英寸(基于屏幕的物理尺寸)。
- px——像素。对应于屏幕上的实际像素。不建议使用这一单位，因为您的用户界面在不同屏幕尺寸的设备上可能不能正确呈现。

这里，将TextView和Button视图的宽度都设置成了一个绝对值。在本例中，TextView的宽度被设置为105dp，Button的宽度被设置为160dp。图3-1展现了在一个分辨率为320×480的模拟器上所观察到的视图效果。

图3-2展现了在一个高分辨率(480×800)的模拟器上所观察到的视图效果。



图 3-1

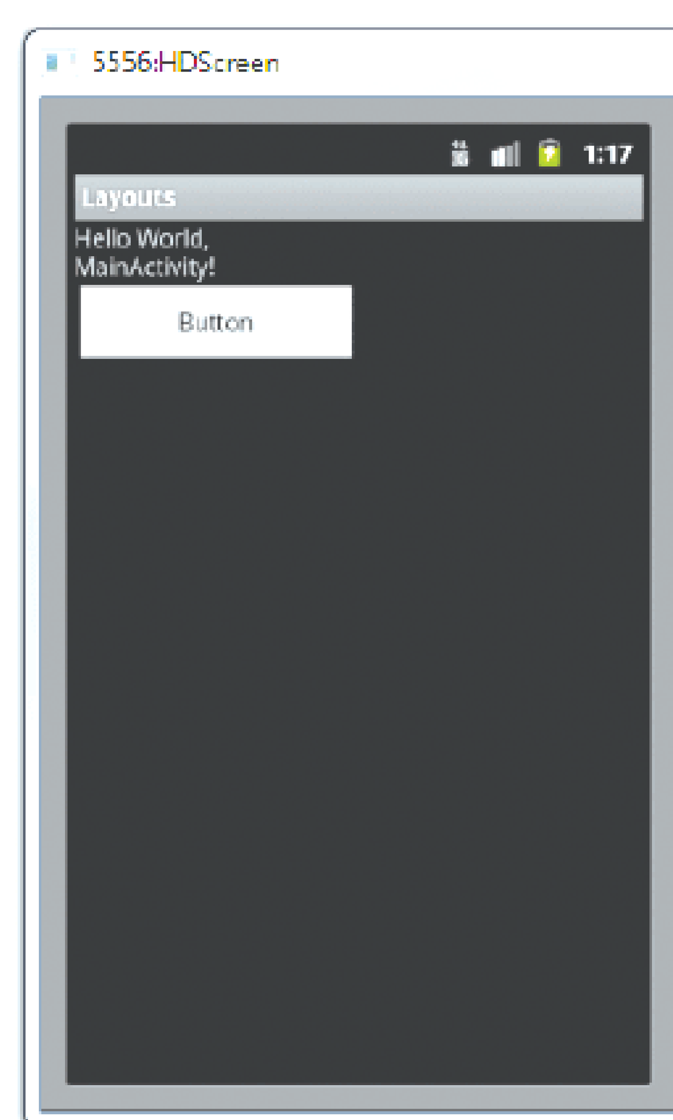


图 3-2

可以看到，在以上两种模拟器中，两种视图的宽度相对于模拟器的宽度而言是一样的。这表明了即使是不同分辨率的目标设备，使用dp作为单位也是有用的，可以确保相对于设备的视图大小保持不变。

上面的例子还指定了布局的方向是垂直的：

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
```

默认的布局方向是水平的。因此，如果省略android:orientation属性，视图将显示如图3-3所示的效果。



图 3-3

在LinearLayout中，可以对其中包含的视图应用layout_weight和layout_gravity属性，可对main.xml文件作如下修改：

```
<? xml version="1.0" encoding="utf-8"? >
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="105dp"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<Button
    android:layout_width="160dp"
    android:layout_height="wrap_content"
    android:text="Button"
    android:layout_gravity="right"
```

```
        android:layout_weight="0.2"
        />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:layout_weight="0.8"
        />
</LinearLayout>
```

图3-4显示了使用layout_gravity属性使按钮与其父元素(指LinearLayout)的右边对齐。同时，使用layout_weight属性指定Button按钮和EditText视图占屏幕剩余空间的比例。所有layout_weight属性的值的和必须为1。

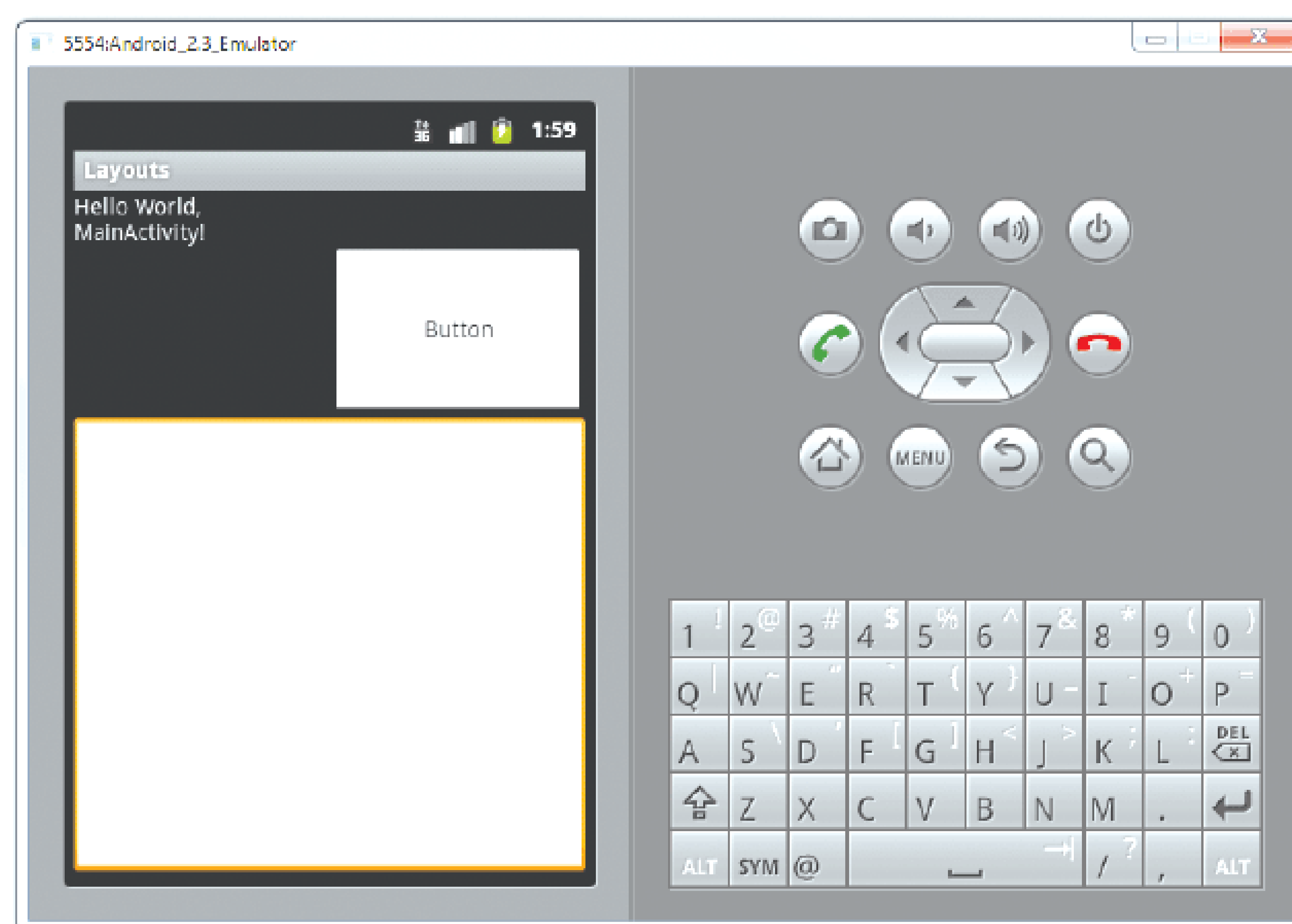


图 3-4

3.1.3　AbsoluteLayout

AbsoluteLayout可用于指定其子元素的确切位置。考虑下列main.xml文件中定义的用户界面：

```
<? xml version="1.0" encoding="utf-8"? >
<AbsoluteLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
<Button
    android:layout_width="188dp"
    android:layout_height="wrap_content"
    android:text="Button"
    android:layout_x="126px"
    android:layout_y="361px"
    />
<Button
    android:layout_width="113dp"
```

```
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_x="12px"
        android:layout_y="361px"
        />
</AbsoluteLayout>
```

图3-5展示了使用android_layout_x和android_layout_y属性将两个Button视图定位在指定的位置处。

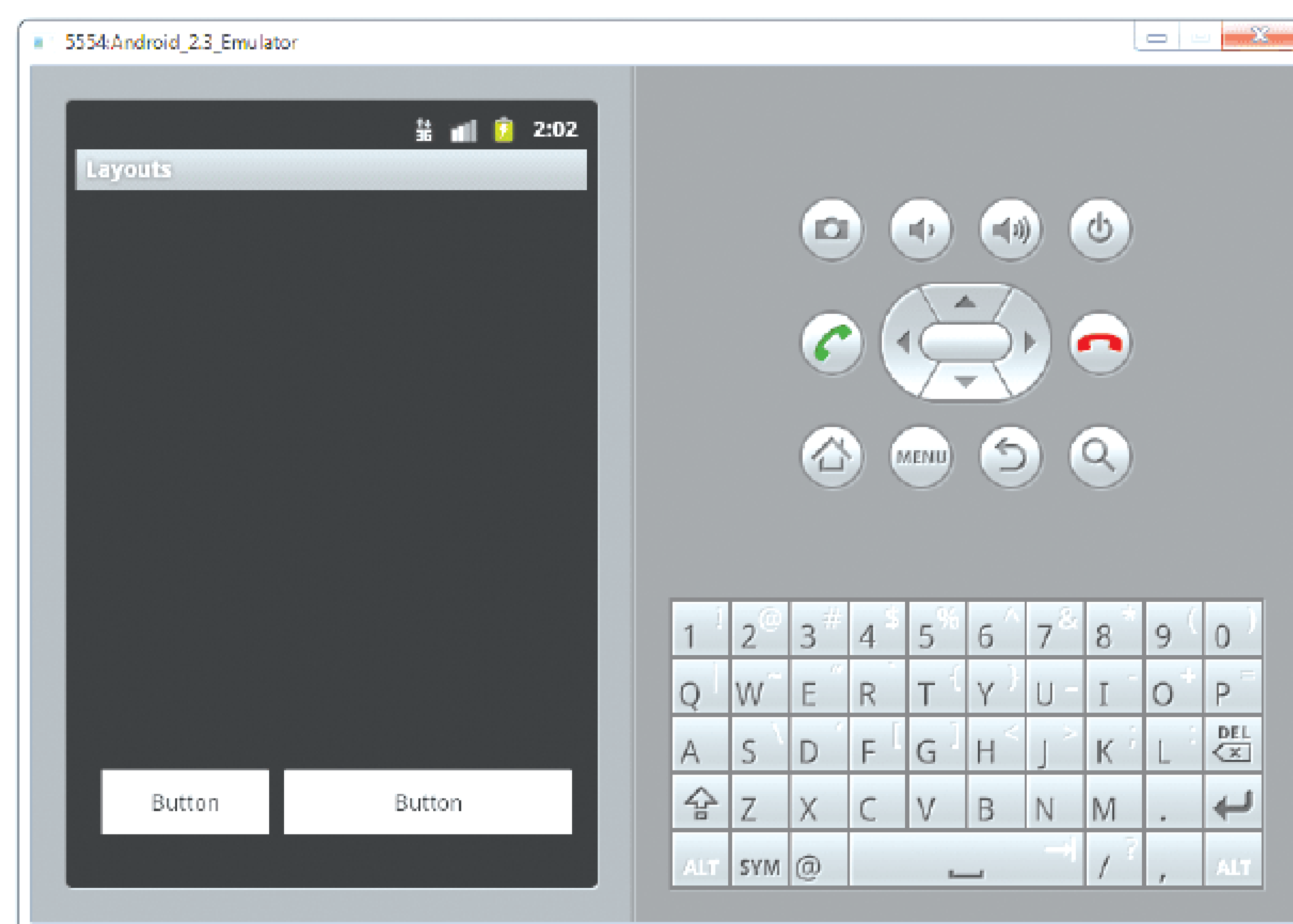


图 3-5

然而，当在高分辨率的屏幕上查看活动时，AbsoluteLayout存在一个问题(如图3-6所示)。因此，从Android 1.5以后，AbsoluteLayout已经被弃用了(尽管当前版本仍旧支持它)。鉴于不能保证在Android的未来版本中是否支持此视图组，应该在用户界面中避免使用AbsoluteLayout，而使用本章描述的其他布局。

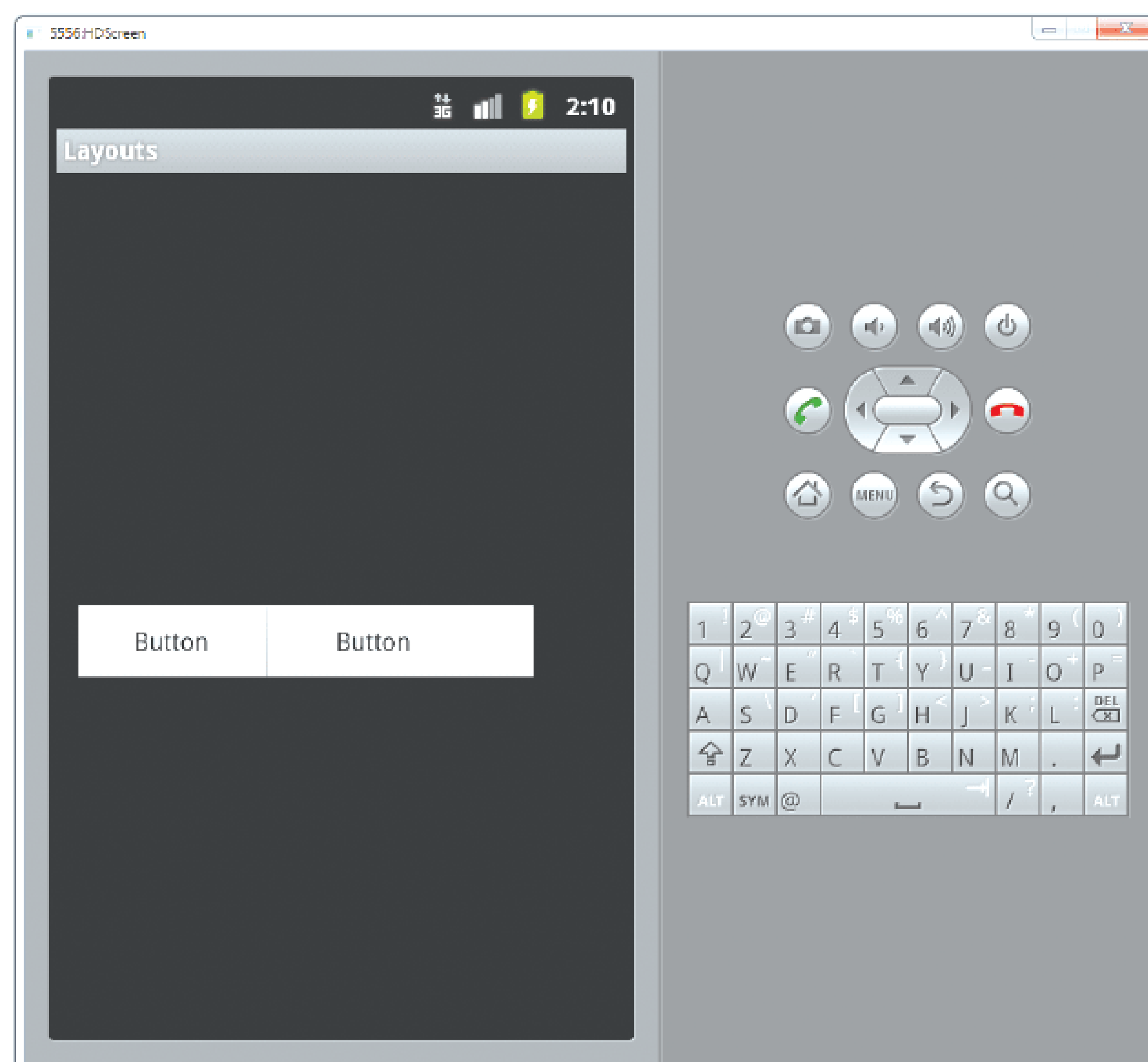


图 3-6

3.1.4　TableLayout

TableLayout以行和列的形式组织视图。使用<TableRow>元素来指定表中的某一行。每一行可以包含一个或多个视图。行内的每个视图构成一个单元格。每一列的宽度由此列中最大单元格的宽度来决定。

考虑下列main.xml文件的内容：

```
<? xml version="1.0" encoding="utf-8"? >
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    >
    <TableRow>
        <TextView
            android:text="User Name:"
            android:width ="120px"
            />
        <EditText
            android:id="@+id/txtUserName"
            android:width="200px" />
    </TableRow>
    <TableRow>
        <TextView
            android:text="Password:"
            />
        <EditText
            android:id="@+id/txtPassword"
            android:password="true"
            />
    </TableRow>
    <TableRow>
        <TextView />
        <CheckBox android:id="@+id/chkRememberPassword"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Remember Password"
            />
    </TableRow>
    <TableRow>
        <Button
            android:id="@+id/buttonSignIn"
            android:text="Log In" />
    </TableRow>
</TableLayout>
```

图3-7展示了以上代码在Android模拟器上呈现的效果。

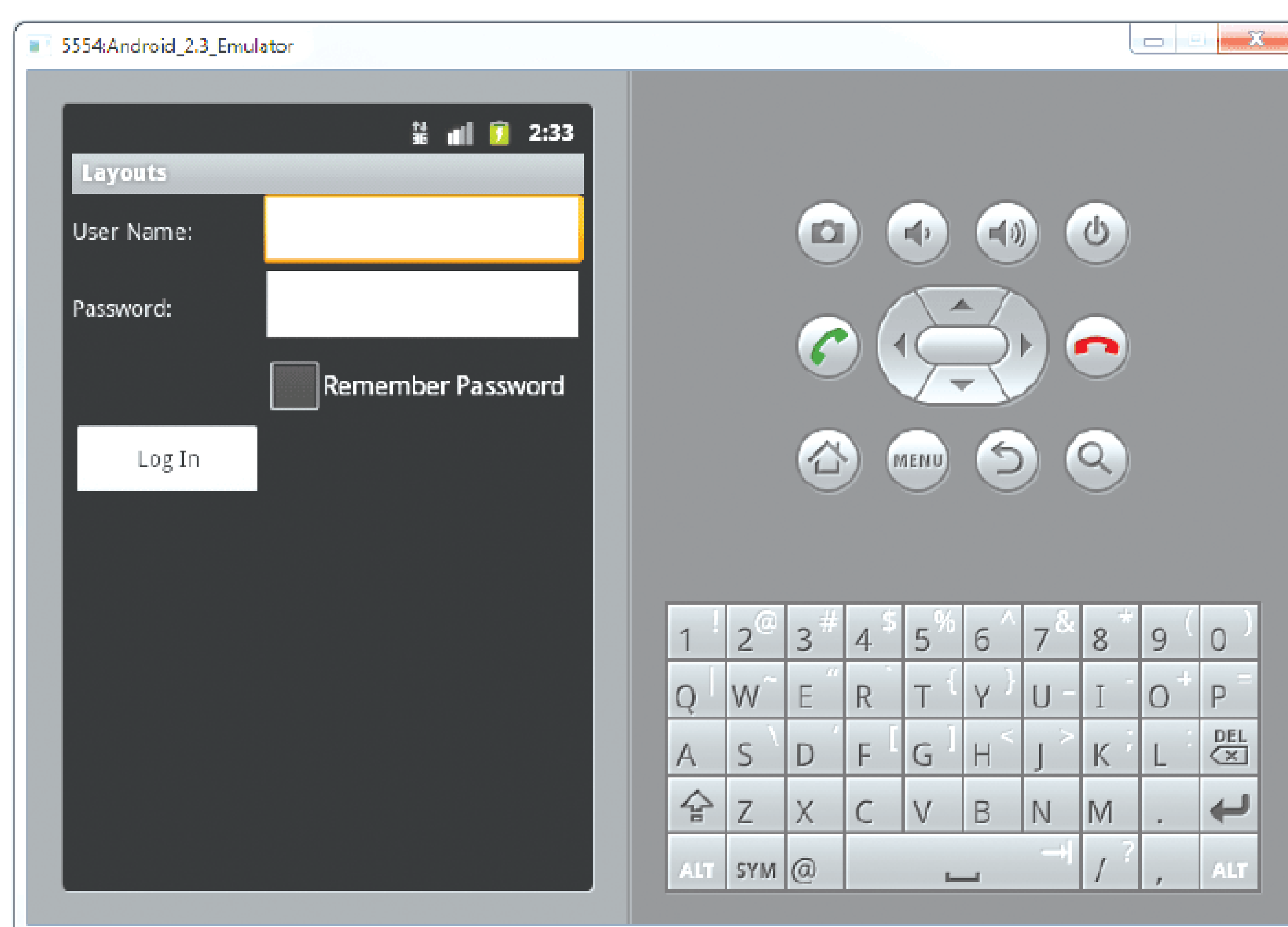


图 3-7

可注意到，在上述例子中，TableLayout中有4行2列。直接位于Password TextView之下的单元格用<TextView/>空元素填充。如果不这样做，Remember Password复选框将显示在Password TextView之下，如图3-8中所示。

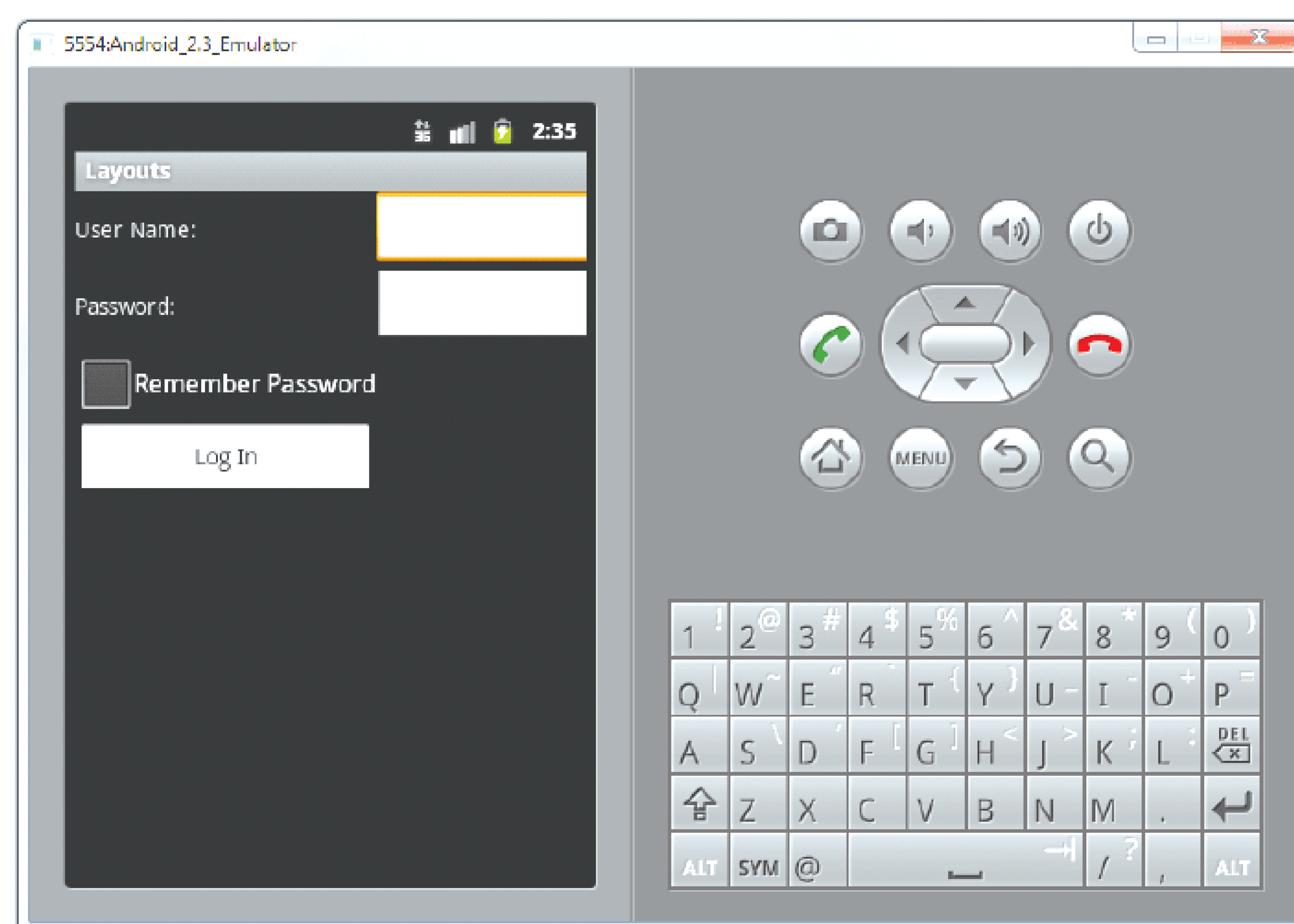


图 3-8

3.1.5 RelativeLayout

RelativeLayout可用于指定子视图相对于彼此之间是如何定位的。考虑下列main.xml文件的内容：

```
<? xml version="1.0" encoding="utf-8"? >
<RelativeLayout
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Comments"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        />
    <EditText
        android:id="@+id/txtComments"
        android:layout_width="fill_parent"
        android:layout_height="170px"
        android:textSize="18sp"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true"
        />
    <Button
        android:id="@+id/btnSave"
        android:layout_width="125px"
        android:layout_height="wrap_content"
        android:text="Save"
        android:layout_below="@+id/txtComments"
        android:layout_alignRight="@+id/txtComments"
        />
    <Button
        android:id="@+id/btnCancel"
        android:layout_width="124px"
        android:layout_height="wrap_content"
        android:text="Cancel"
        android:layout_below="@+id/txtComments"
        android:layout_alignLeft="@+id/txtComments"
        />
</RelativeLayout>
```

可注意到，每一个嵌入RelativeLayout中的视图都有使它与其他视图对齐的属性。这些属性如下所示：

- layout_alignParentTop
- layout_alignParentLeft
- layout_alignLeft
- layout_alignRight
- layout_below
- layout_centerHorizontal

每一个属性的值是引用的视图的ID。前面的XML用户界面创建的屏幕如图3-9所示。

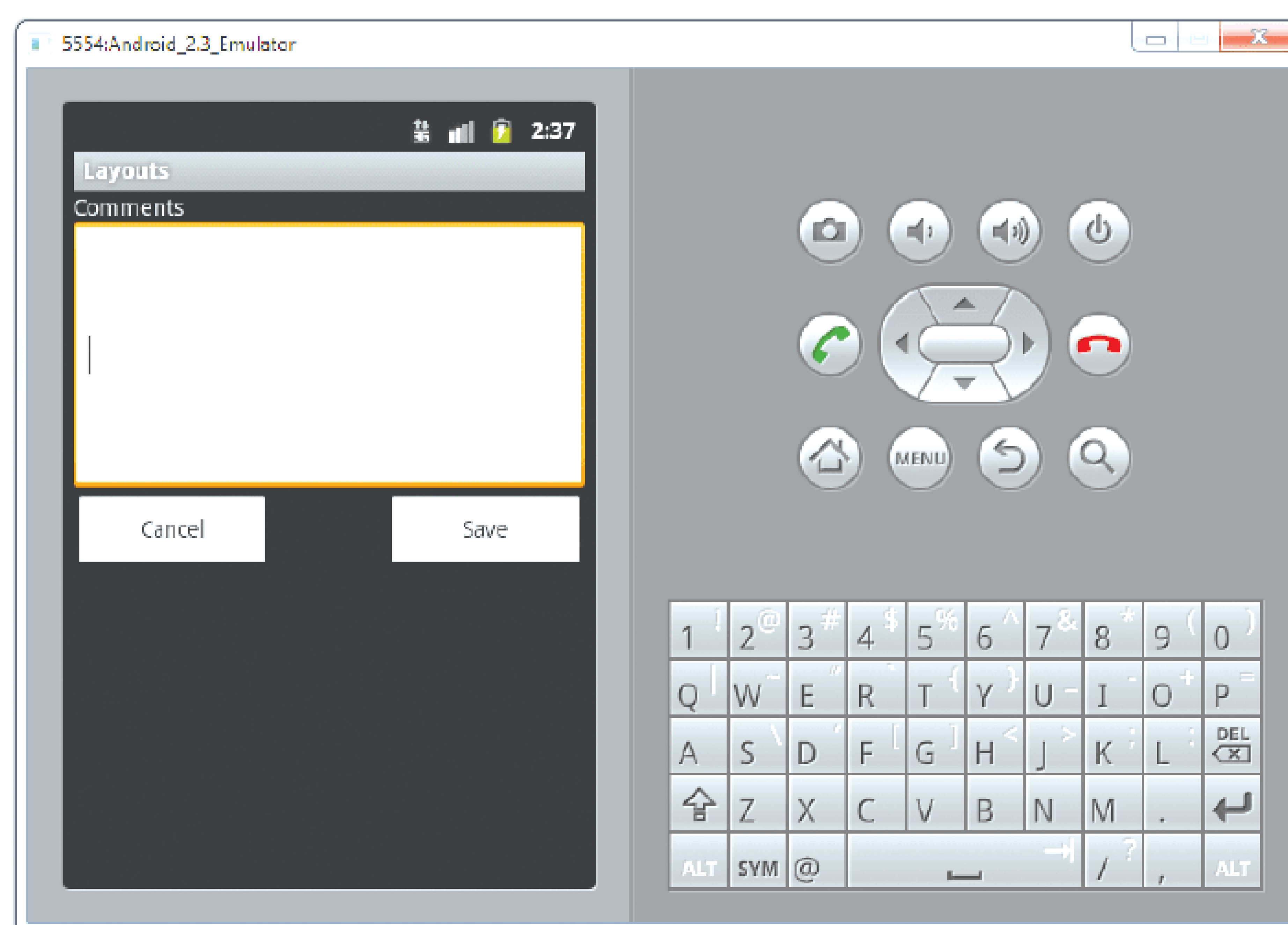


图 3-9

3.1.6 FrameLayout

FrameLayout是一个在屏幕上可以用来显示单个视图的占位符。添加到FrameLayout中的视图常常锚定在布局的左上方。考虑main.xml文件中的下列内容:

```
<? xml version="1.0" encoding="utf-8"? >
<RelativeLayout
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is my lovely dog, Ookii"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        />
    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true"
        >
        <ImageView
            android:src = "@drawable/ookii"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            />
        </FrameLayout>
    </RelativeLayout>
```

这里，RelativeLayout中有一个FrameLayout。在该FrameLayout中嵌入了一个ImageView。用户界面如图3-10所示。

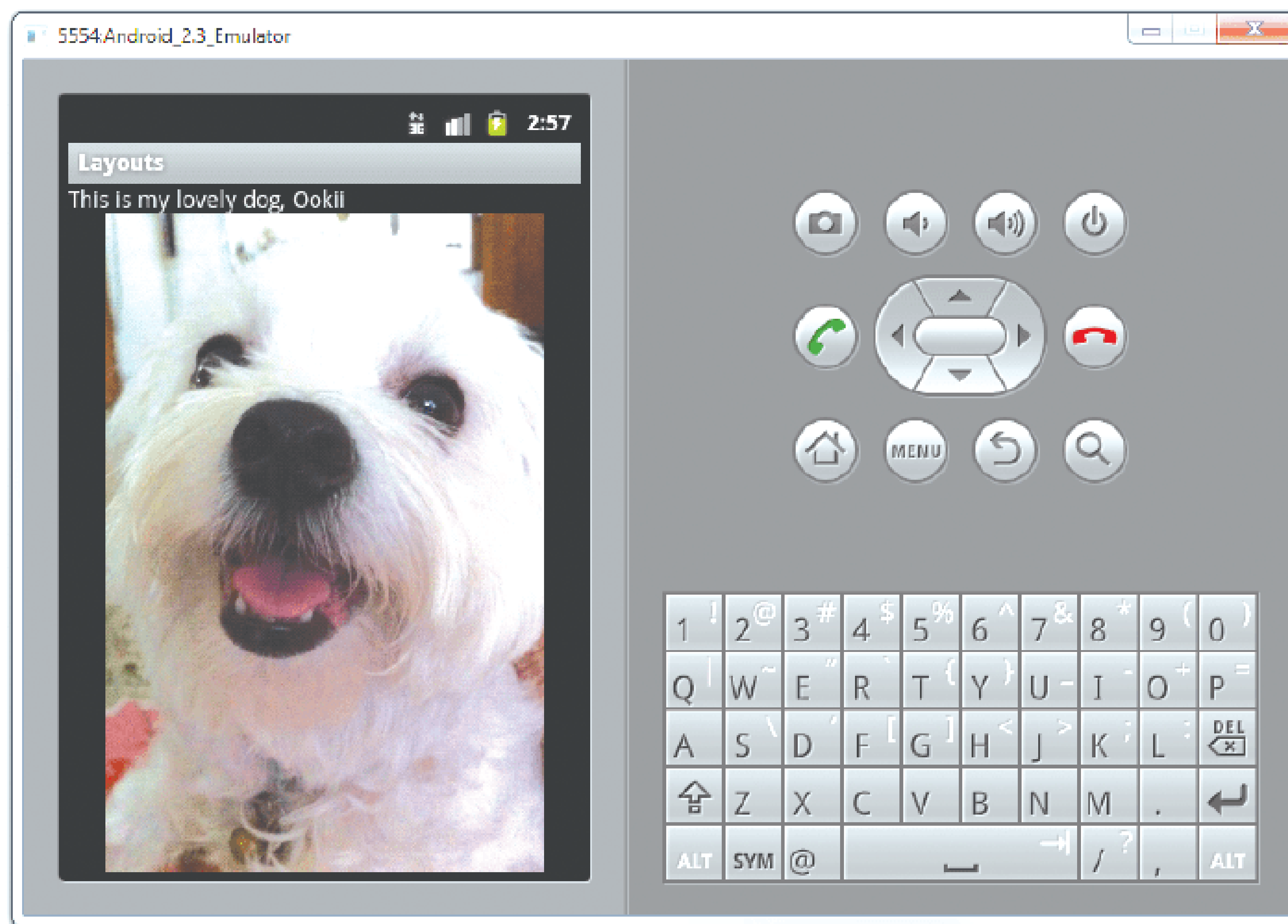
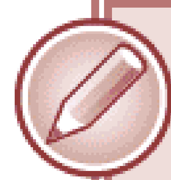


图 3-10



注意：这个例子假设res/drawable-mdpi文件夹下有一个名为ookii.png的图像。

如果在FrameLayout中添加另一个视图(如Button视图)，这个视图将覆盖先前的视图(如图3-11所示)：

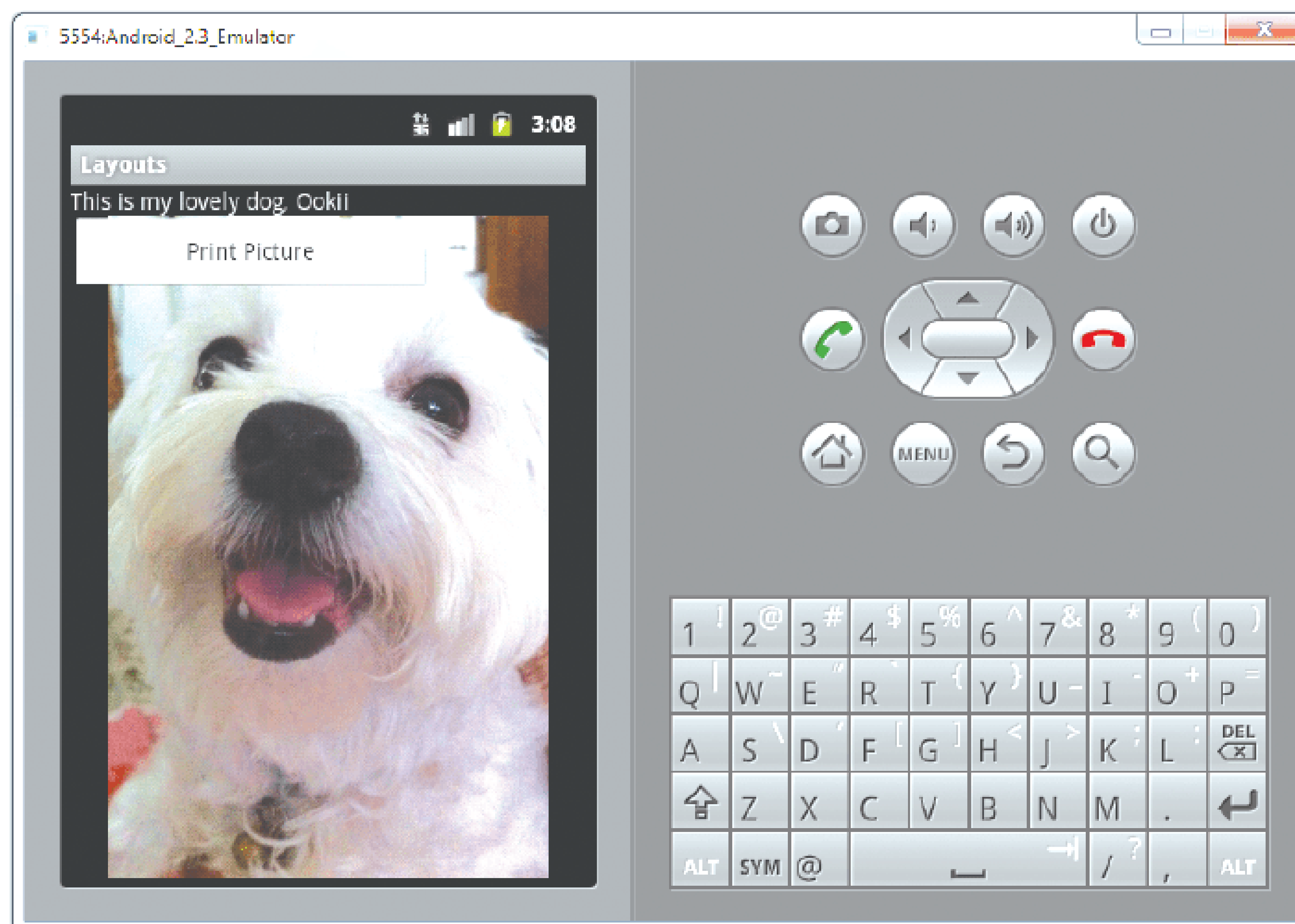


图 3-11

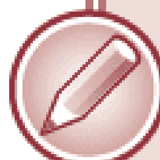
```
<? xml version="1.0" encoding="utf-8"? >
<RelativeLayout
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is my lovely dog, Ookii"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        />
    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true"
        >
        <ImageView
            android:src = "@drawable/ookii"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            />
        <Button
            android:layout_width="124dp"
            android:layout_height="wrap_content"
            android:text="Print Picture"
            />
    </FrameLayout>
</RelativeLayout>
```



注意：虽然可以在FrameLayout中添加多个视图，但每一个视图都堆叠在前一个上面。这在想将一系列图像变成动画，使得每次只有一个视图可见的情况下很有用。

3.1.7 ScrollView

ScrollView是一种特殊类型的FrameLayout，因为它可以使用户滚动显示一个占据的空间大于物理显示的视图列表。ScrollView只能包含一个子视图或视图组，通常是LinearLayout。



注意：不要将ScrollView和ListView(第4章将讨论)一起使用。ListView用来显示一个相关信息的列表并针对大列表的处理进行了优化。

下面的main.xml的内容显示了一个包含LinearLayout的ScrollView，而LinearLayout又包含了一些Button和EditText视图：

```
<? xml version="1.0" encoding="utf-8"? >
<ScrollView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        >
        <Button
            android:id="@+id/button1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 1"
            />
        <Button
            android:id="@+id/button2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 2"
            />
        <Button
            android:id="@+id/button3"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 3"
            />
        <EditText
            android:id="@+id/txt"
            android:layout_width="fill_parent"
            android:layout_height="300px"
            />
        <Button
            android:id="@+id/button4"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 4"
            />
        <Button
            android:id="@+id/button5"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 5"
            />
    </LinearLayout>
</ScrollView>
```

图3-12展示了ScrollView可以使用户向上拖动屏幕以显示位于屏幕底部的视图。

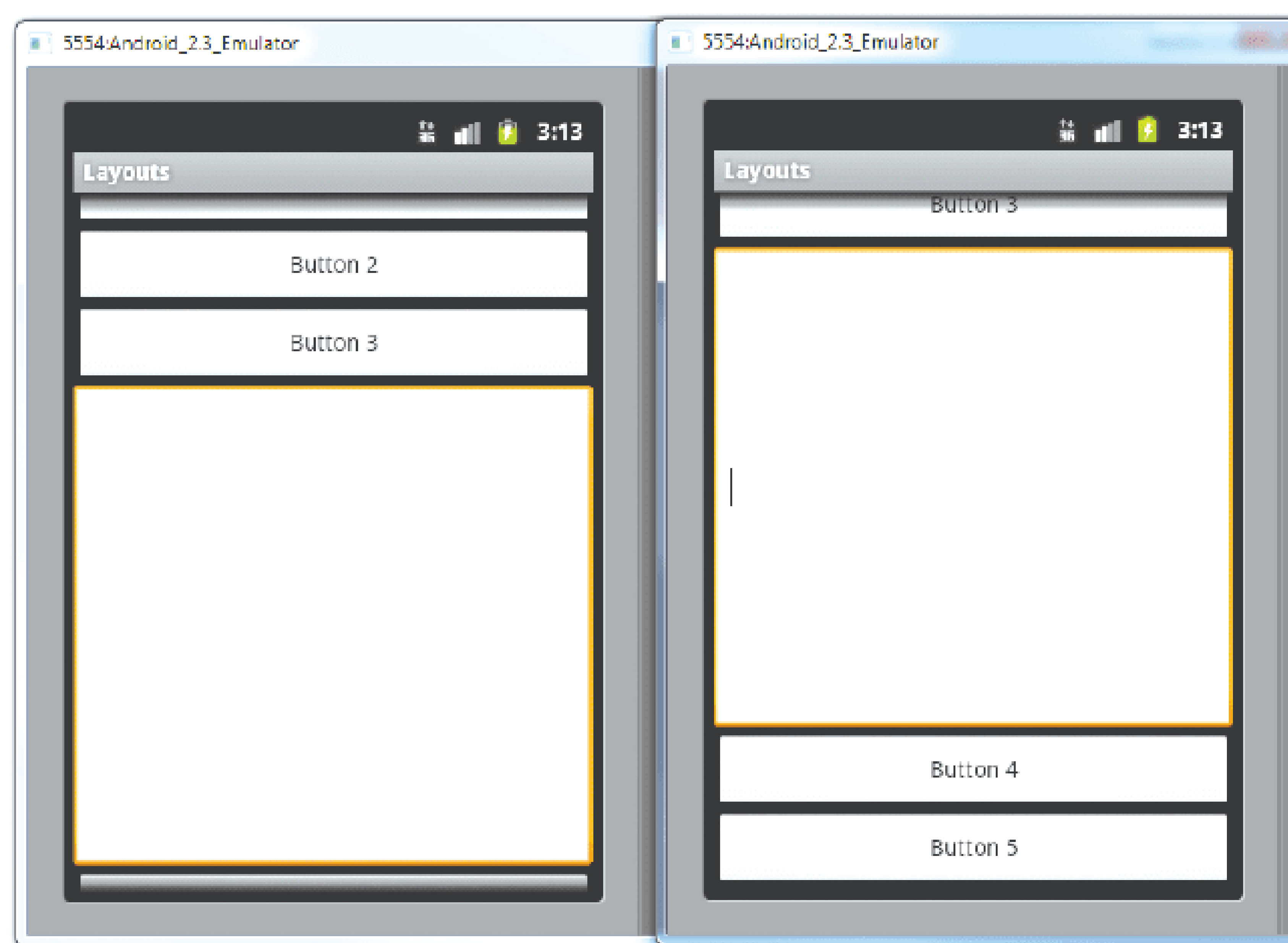
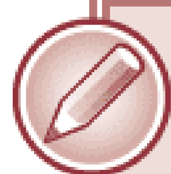


图 3-12

3.2 适应显示方向

现代智能手机的主要特征之一是它们切换屏幕方向的能力，Android也不例外。Android支持两种屏幕方向：纵向和横向。默认情况下，当改变Android设备的显示方向时，当前显示的活动将自动在新方向上重绘其内容。这是因为当显示方向上发生改变时，都会触发活动的onCreate()事件。



注意： 当改变Android设备的方向时，当前活动实际上是先被销毁，然后再重新创建。

然而，当重绘时，视图可能会按照其原始位置绘制(这取决于所选择的布局)。图3-13展示了之前提到的一个例子，分别以纵向和横向模式进行显示。

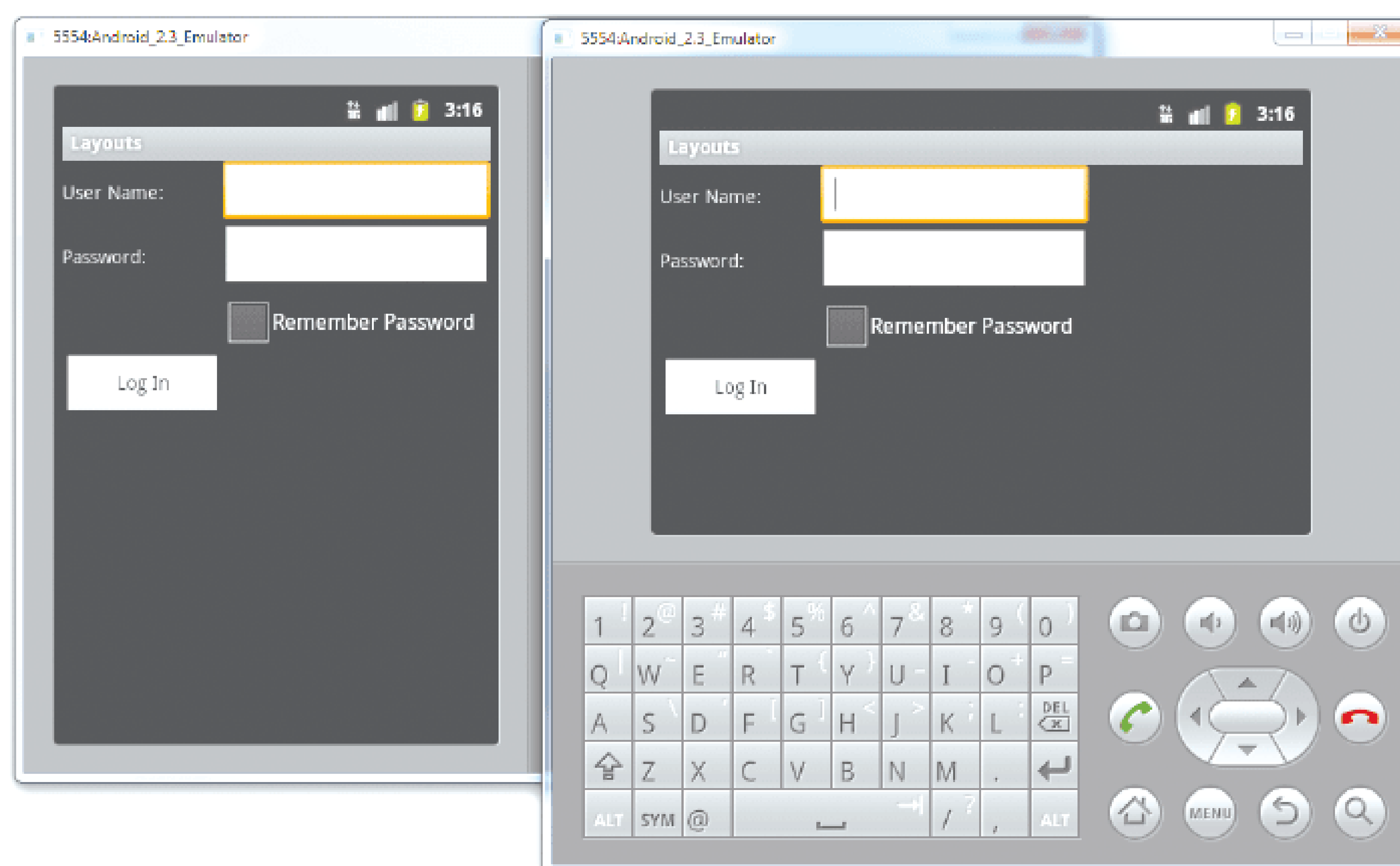


图 3-13

在横向模式下可以看到，屏幕右侧有大量的空余空间可以使用。而且，当屏幕方向被设置成横向时，任何位于屏幕底部的额外的视图将被隐藏。

通常，可以使用如下两种技术来处理屏幕方向的变化：

- 锚定——将视图锚定到屏幕的四条边是最容易的方法。当屏幕方向改变时，视图可以整齐地锚定于屏幕边缘。
- 调整大小和重新定位——尽管锚定和居中显示的技术简单，可以确保视图能处理屏幕方向的变化，但最佳的技术还是根据当前屏幕方向重新调整每一个视图的大小。

3.2.1　锚定视图

使用RelativeLayout可以很容易实现锚定。考虑下列main.xml文件，其中包含了嵌入在<RelativeLayout>元素中的5个Button视图：

```
<? xml version="1.0" encoding="utf-8"? >
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Left Button"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Right Button"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Left Button"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"
        />
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Right Button"
```

```
            android:layout_alignParentRight="true"
            android:layout_alignParentBottom="true"
            />
        <Button
            android:id="@+id/button5"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Middle Button"
            android:layout_centerVertical="true"
            android:layout_centerHorizontal="true"
            />
    </RelativeLayout>
```

注意不同Button视图中所具有的以下属性：

- layout_alignParentLeft——将视图与其父视图的左侧对齐
- layout_alignParentRight——将视图与其父视图的右侧对齐
- layout_alignParentTop——将视图与其父视图的上部对齐
- layout_alignParentBottom——将视图与其父视图的底部对齐
- layout_centerVertical——使视图在其父视图中垂直居中
- layout_centerHorizontal——使视图在其父视图中水平居中

图3-14展示了以纵向模式观察到的活动的效果。

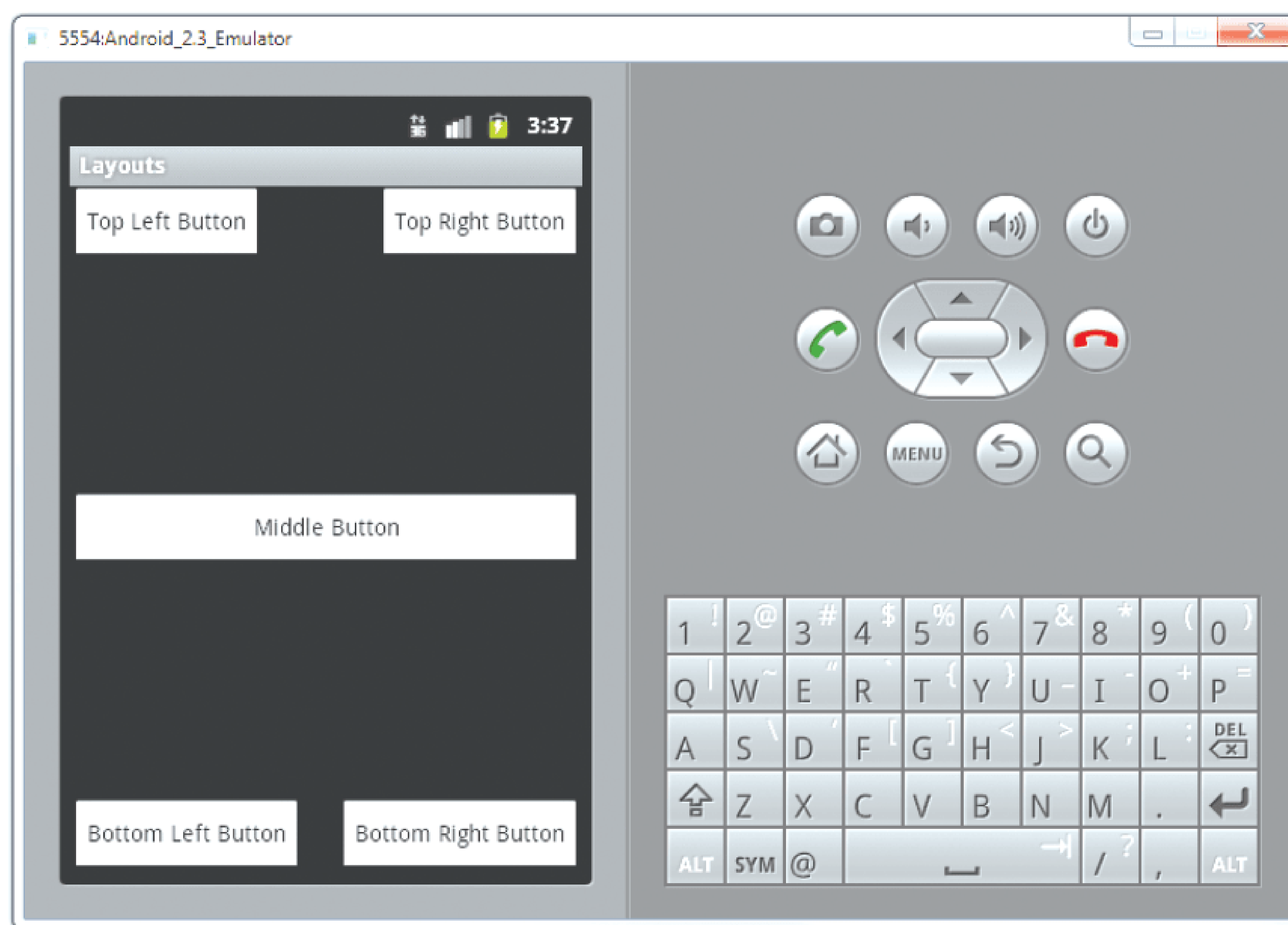


图 3-14

当屏幕方向改变为横向模式时，4个按钮对齐到屏幕的四边。中间的按钮在屏幕中央显示，宽度完全拉伸到整个屏幕(如图3-15所示)。

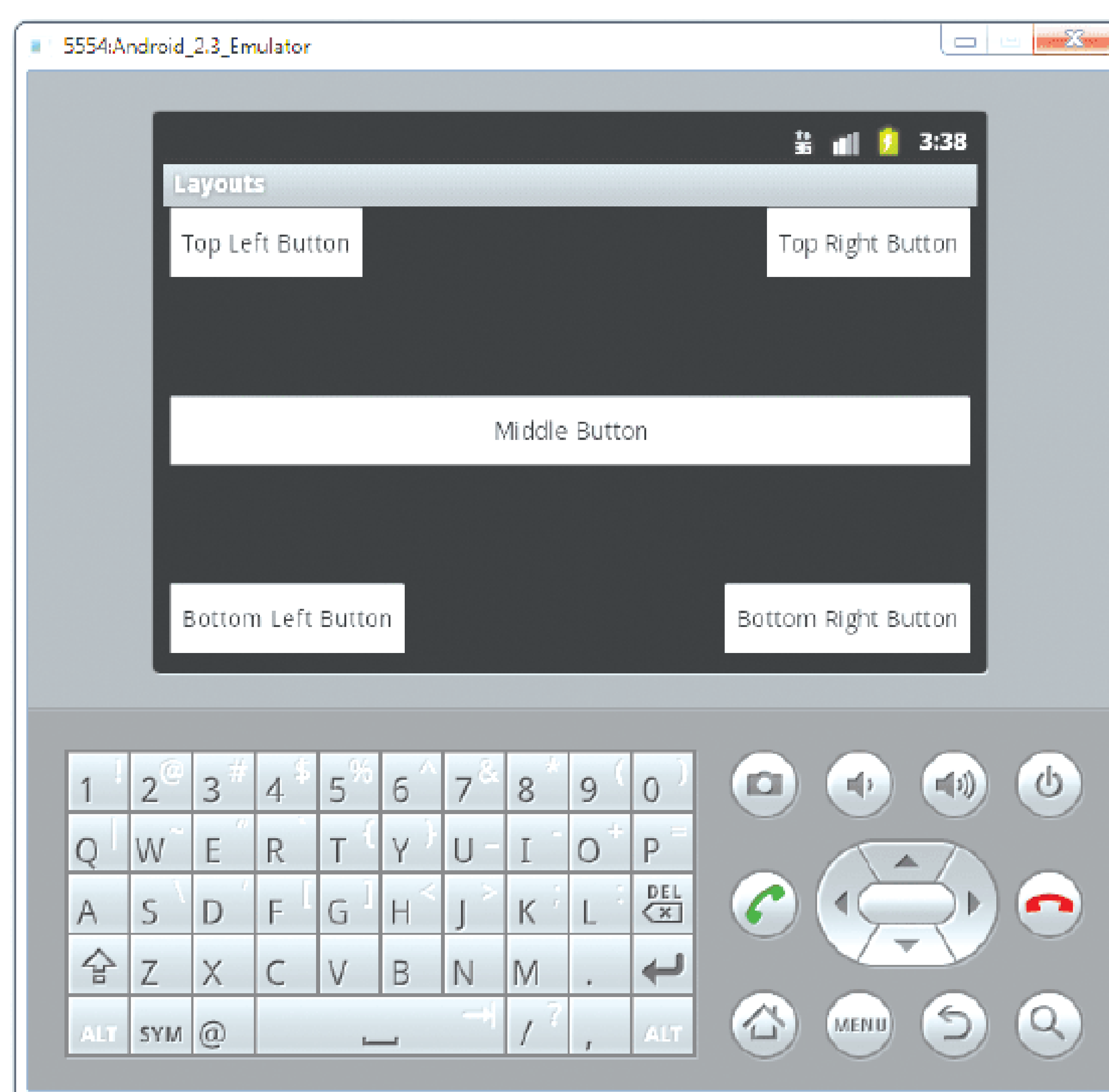


图 3-15

3.2.2　调整大小和重新定位

除了将视图锚定到屏幕的四边之外，基于屏幕方向定制用户界面的更简单的方法是为每个方向上的用户界面创建一个包含XML文件的单独的res/layout文件夹。为了支持横向模式，可以在res文件夹下创建一个新文件夹，并命名为layout-land(表示横向)。图3-16显示了这样的含有main.xml文件的新文件夹。

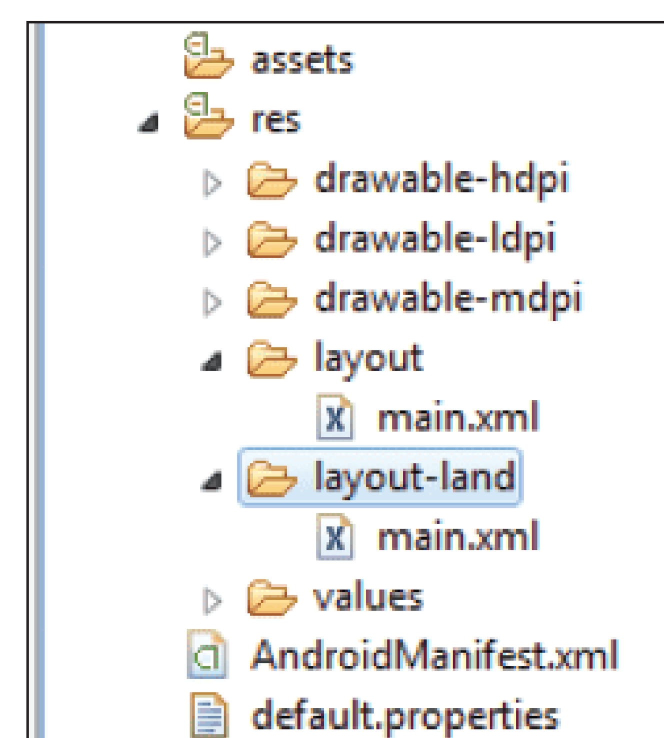


图 3-16

基本上，包含在layout文件夹下的main.xml文件为活动定义了纵向模式的用户界面，而在layout-land文件夹下的main.xml文件定义了横向模式的用户界面。

layout文件夹下的main.xml文件的内容如下所示：

```
<? xml version="1.0" encoding="utf-8"? >
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Left Button"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        />
    <Button
        android:id="@+id/button2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Right Button"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Left Button"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"
        />
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Right Button"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        />
    <Button
        android:id="@+id/button5"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Middle Button"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        />
</RelativeLayout>
```

下面显示了layout-land文件夹下的main.xml文件的内容(粗体显示的语句是以横向模式显示的额外视图):

```
<? xml version="1.0" encoding="utf-8"? >
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Left Button"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        />
    <Button
```

```
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Right Button"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Left Button"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"
        />
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Right Button"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        />
    <Button
        android:id="@+id/button5"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Middle Button"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        />
    <Button
        android:id="@+id/button6"
        android:layout_width="180px"
        android:layout_height="wrap_content"
        android:text="Top Middle Button"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:layout_alignParentTop="true"
        />
    <Button
        android:id="@+id/button7"
        android:layout_width="180px"
        android:layout_height="wrap_content"
        android:text="Bottom Middle Button"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        />
</RelativeLayout>
```

当活动以纵向模式加载时，将显示5个按钮，如图3-17所示。

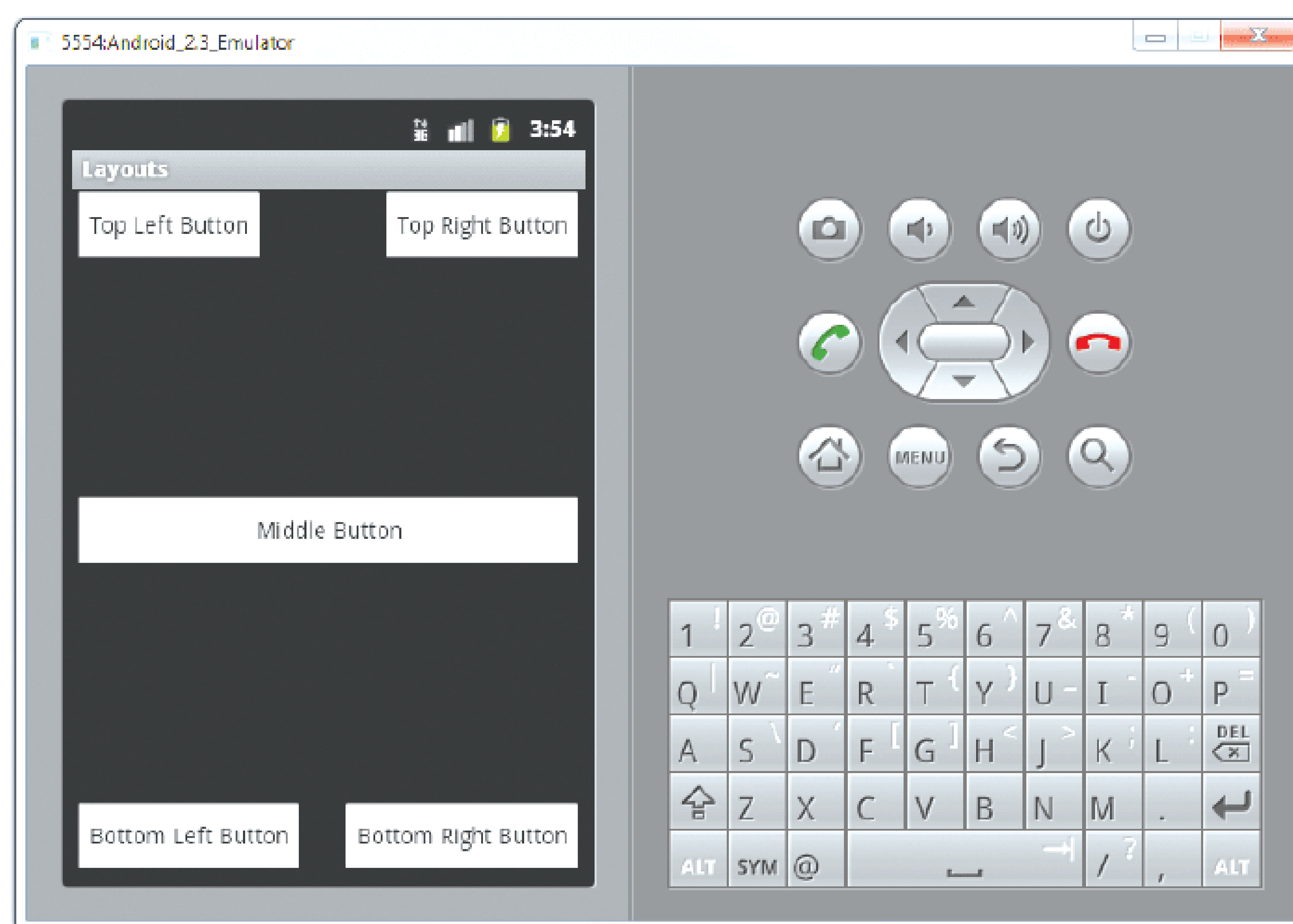


图 3-17

当活动以横向模式加载时，将出现7个按钮(如图3-18所示)，这证明当设备处于不同方向时，将加载不同的XML文件。



图 3-18

使用这种方法，当设备方向改变时，Android将根据当前屏幕方向为活动自动加载相应的XML文件。

3.3 管理屏幕方向的变化

既然已经了解了如何实现两种技术以便适应屏幕方向的变化，那么让我们探究一下当设备改变方向时活动的状态会发生什么情况。下面的“试一试”展示了当设备改变方向时活动的行为。

试一试　了解方向改变时活动的行为

Orientations.zip代码文件可以在Wrox.com上下载

(1) 使用Eclipse，创建一个名为Orientations的新项目。

(2) 在main.xml文件中添加下列粗体显示的语句：

```
<? xml version="1.0" encoding="utf-8"? >
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<EditText
    android:id="@+id/txtField1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
</LinearLayout>
```

(3) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Orientations;

import android.app.Activity;
import android.os.Bundle;

import android.util.Log;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Log.d("StateInfo", "onCreate");
    }

    @Override
    public void onStart() {
        Log.d("StateInfo", "onStart");
        super.onStart();
    }

    @Override
    public void onResume() {
        Log.d("StateInfo", "onResume");
        super.onResume();
```

```
        }

        @Override
        public void onPause() {
            Log.d("StateInfo", "onPause");
            super.onPause();
        }

        @Override
        public void onStop() {
            Log.d("StateInfo", "onStop");
            super.onStop();
        }

        @Override
        public void onDestroy() {
            Log.d("StateInfo", "onDestroy");
            super.onDestroy();
        }

        @Override
        public void onRestart() {
            Log.d("StateInfo", "onRestart");
            super.onRestart();
        }
    }
```

(4) 按F11键在Android模拟器上调试应用程序。

(5) 在两个EditText视图中输入一些文本(如图3-19所示)。

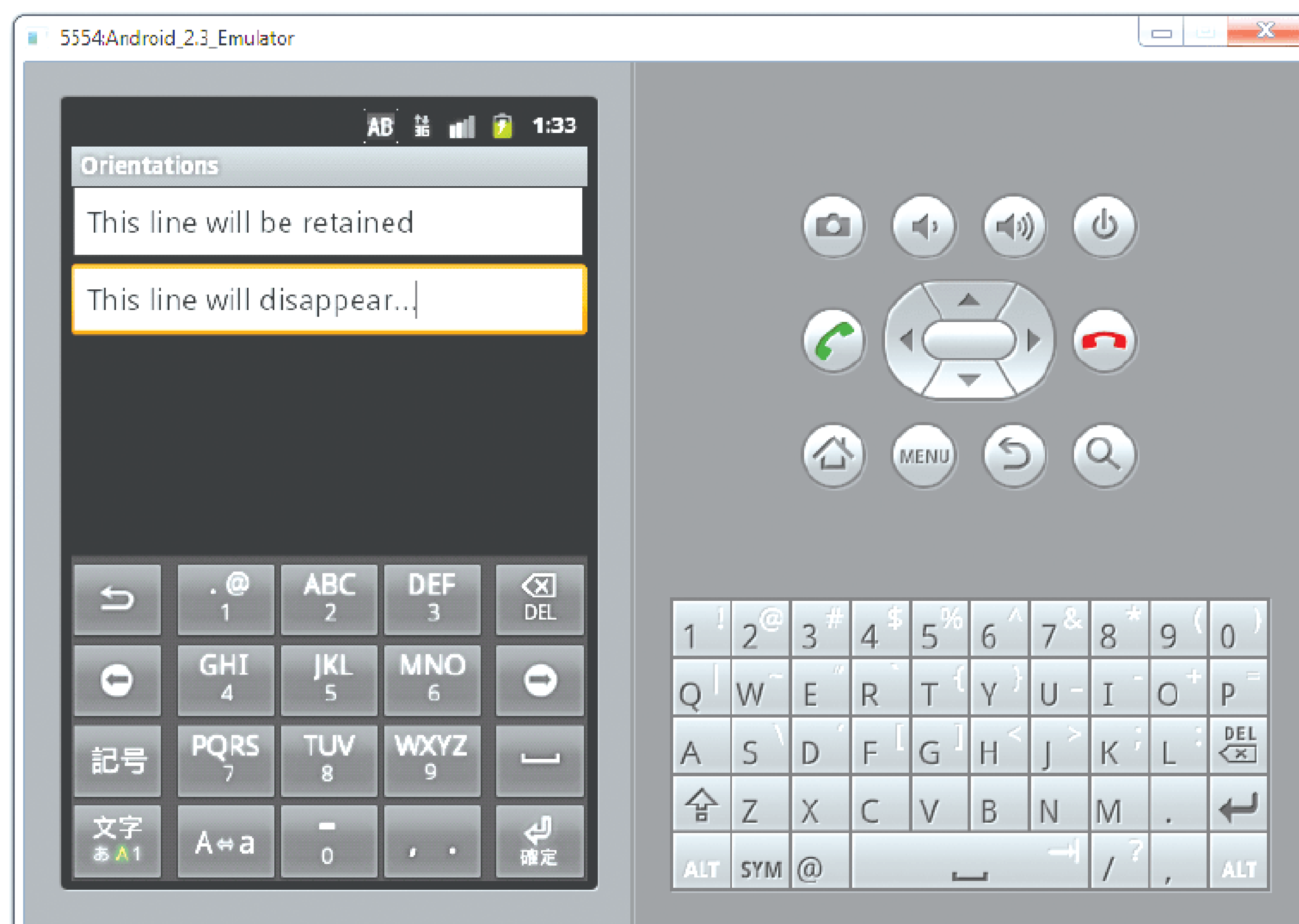


图 3-19

(6) 按Ctrl+F11组合键改变Android模拟器的显示方向。图3-20显示了横向模式下的模拟器效果。注意，第1个EditText视图中的文本仍旧可见，而第2个EditText视图为空。

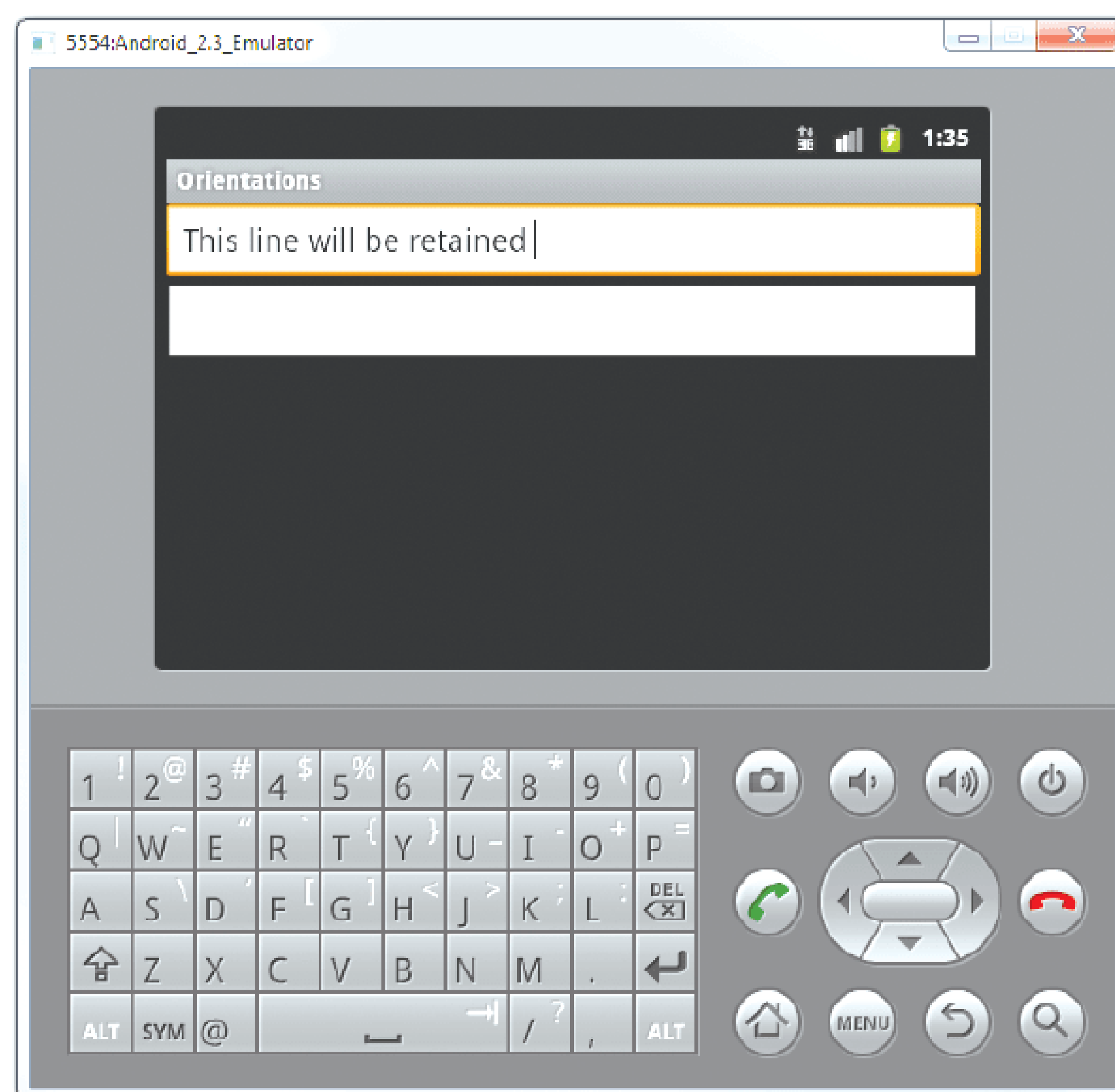


图 3-20

(7) 观察LogCat窗口中的输出(需要在Eclipse中切换到Debug透视图)，应该看到类似如下所示内容：

```
01-05 13:32:30.266: DEBUG/StateInfo(5477): onCreate
01-05 13:32:30.296: DEBUG/StateInfo(5477): onStart
01-05 13:32:30.296: DEBUG/StateInfo(5477): onResume
...
01-05 13:35:20.106: DEBUG/StateInfo(5477): onPause
01-05 13:35:20.106: DEBUG/StateInfo(5477): onStop
01-05 13:35:20.106: DEBUG/StateInfo(5477): onDestroy
01-05 13:35:20.246: DEBUG/StateInfo(5477): onCreate
01-05 13:35:20.256: DEBUG/StateInfo(5477): onStart
01-05 13:35:20.256: DEBUG/StateInfo(5477): onResume
```

示例说明

从LogCat窗口中的输出内容可以明显看出，当设备改变方向时，活动先被销毁：

```
01-05 13:35:20.106: DEBUG/StateInfo(5477): onPause
01-05 13:35:20.106: DEBUG/StateInfo(5477): onStop
01-05 13:35:20.106: DEBUG/StateInfo(5477): onDestroy
```

然后，它被重建：

```
01-05 13:35:20.246: DEBUG/StateInfo(5477): onCreate
01-05 13:35:20.256: DEBUG/StateInfo(5477): onStart
01-05 13:35:20.256: DEBUG/StateInfo(5477): onResume
```

了解这一行为是重要的，因为需要确保采取必要的措施来保持方向改变之前活动的状态。

例如，在活动中有包含一些计算所需的值的变量。对于任意活动，应该在onPause()事件中保存任何您需要保存的状态，这一事件在每次活动改变方向时触发。3.3.1节讲述保存状态信息的不同方法。

另一个需要理解的重要行为是当包含视图的活动被销毁时，只有那些在这个活动中被命名的视图(通过android:id属性)才能保持它们的状态。例如，在向EditText视图中输入一些文本同时，用户可能会改变显示方向。出现这种情况时，EditText视图中的任何文本将被保持并在活动重新创建时自动恢复。相反，如果没有使用android:id属性命名EditText视图，活动将无法保持视图中当前所含文本。

3.3.1 配置改变时保持状态信息

到目前为止，您已经学习了屏幕方向改变时会销毁和重建一个活动。记住，当重建一个活动时，活动的当前状态可能会丢失。当终止一个活动时，将触发以下两个事件中的一个或多个：

- onPause()——当一个活动被终止或转入后台时都会触发这一事件。
- onSaveInstanceState()——当一个活动将要被终止或转入后台时，也会触发这一事件(正如onPause()事件一样)。然而，与onPause()事件不同的是，当一个活动从栈中卸载时(例如用户按下了Back按钮)不会触发onSaveInstanceState()事件，这是因为后面无须恢复其状态。

简而言之，要保持一个活动的状态，可以实现onPause()事件，然后使用自己的方法来保存活动状态，如利用数据库、内部或外部的文件存储器等。

如果只是想保存活动状态用来在以后活动重建时(例如当设备改变方向时)进行恢复，那么更简单的方法是实现onSaveInstanceState()方法。这个方法提供了Bundle对象作为一个参数，可以用它保存活动的状态。以下代码展示了在onSaveInstanceState事件中可以将字符串ID保存到Bundle对象中：

```
@Override
public void onSaveInstanceState(Bundle outState) {
    //---保存您想保持的任何状态---
    outState.putString("ID", "1234567890");
    super.onSaveInstanceState(outState);
}
```

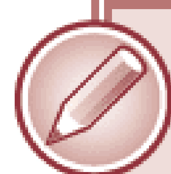
当重建一个活动时，首先触发onCreate()事件，随后是onRestoreInstanceState()事件。此事件可以使您检索先前在onSaveInstanceState事件中通过其参数Bundle对象保存的状态：

```
@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    //---检索先前保留的信息---
    String ID = savedInstanceState.getString("ID");
```

尽管可以使用onSaveInstanceState()事件来保存状态信息，但要注意只能通过一个Bundle对象保存状态信息的局限性。如果需要保存更复杂的数据结构，这就不是一个合适的解决方法。

另一个可以使用的事件处理程序是onRetainNonConfigurationInstance()事件。当一个活动由于配置改变将要被销毁时会触发这一事件。可以通过在该事件中返回来保存当前的数据，如下所示：

```
@Override
    public Object onRetainNonConfigurationInstance() {
        //---在此处保存任何您想要保存的数据；它接受Object类型---
        return("Some text to preserve");
    }
```



注意：当屏幕方向改变时，这一变化是所谓配置改变的一部分。配置改变将销毁您的当前活动。

注意，此事件返回一个Object类型，它几乎允许返回任何数据类型。要提取保存的数据，可以使用getLastNonConfigurationInstance()方法在onCreate()事件中进行提取，如下所示：

```
@Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Log.d("StateInfo", "onCreate");
        String str = (String) getLastNonConfigurationInstance();
    }
```

3.3.2 检测方向改变

有时需要在运行时知道设备的当前方向。要确定这一点，可以使用WindowManger类。以下代码片段演示了如何以编程方式来检测活动的当前方向：

```
import android.util.Log;
import android.view.Display;
import android.view.WindowManager;
//...
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //---获取当前显示信息---
        WindowManager wm = getWindowManager();
        Display d = wm.getDefaultDisplay();

        if (d.getWidth() > d.getHeight())
        {
            //---横向模式---
            Log.d("Orientation", "Landscape mode");
        }
        else
        {
            //---纵向模式---
            Log.d("Orientation", "Portrait mode");
        }
    }
```

getDefaultDisplay()方法返回一个表示设备屏幕的Display对象。然后，您可以获得其宽度和高度，进而推断出当前的屏幕方向。

3.3.3 控制活动的方向

有时，您也许想要保证应用程序只在一个特定的方向上显示。例如，您可能正在编写一个只能以横向模式呈现的游戏。在这种情况下，可以使用Activity类的setRequestOrientation()方法以编程方式强制改变显示方向：

```
import android.content.pm.ActivityInfo;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //---改为横向模式---
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
}
```

要改为纵向模式，可使用ActivityInfo.SCREEN_ORIENTATION_PORTRAIT常量：

```
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
```

除了使用setRequestOrientation()方法，还可以在AndroidManifest.xml文件中的<activity>元素上使用android:screenOrientation属性，来将活动限制在一个特定方向上，如下所示：

```
<? xml version="1.0" encoding="utf-8"? >
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Orientations"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name"
                  android:screenOrientation="landscape" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
</manifest>
```

上述的例子将活动限制在一个特定的方向上(此处是横向)，并且防止活动被销毁。也就是说，当设备方向改变时，活动不会被销毁并且也不会再次触发onCreate()事件。

以下是在android:screenOrientation属性中可以指定的两个值：

- portrait——纵向模式
- sensor——基于加速计

3.4　以编程方式创建用户界面

到目前为止，您在本章中所看到的所有用户界面都是使用XML创建的。如前所述，除了使用XML，还可以使用代码创建用户界面。如果用户界面需要在运行时动态生成，这个方法就很有用。例如，设想您正在构建一个电影票预订系统，您的应用程序将使用按钮显示每场电影的座位。这种情况下，就需要根据用户选择的电影来动态生成用户界面。

下面的“试一试”演示了在活动中动态构建用户界面所需的代码。

试一试　通过代码创建用户界面

UICode.zip代码文件可以在Wrox.com上下载

(1) 使用Eclipse，创建一个名为UICode的新Android项目。

(2) 在MainActivity.java文件中，添加下列粗体显示的语句：

```
package net.learn2develop.UICode;

import android.app.Activity;
import android.os.Bundle;

import android.view.ViewGroup.LayoutParams;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.main);

        //---视图的参数---
        LayoutParams params =
            new LinearLayout.LayoutParams(
                LayoutParams.FILL_PARENT,
                LayoutParams.WRAP_CONTENT);

        //---创建一个布局---
        LinearLayout layout = new LinearLayout(this);
        layout.setOrientation(LinearLayout.VERTICAL);

        //---创建一个文本视图---
        TextView tv = new TextView(this);
        tv.setText("This is a TextView");
        tv.setLayoutParams(params);

        //---创建一个按钮---
```

```
        Button btn = new Button(this);
        btn.setText("This is a Button");
        btn.setLayoutParams(params);

        //---添加文本视图---
        layout.addView(tv);

        //---添加按钮---
        layout.addView(btn);

        //---为布局创建一个布局参数---
        LinearLayout.LayoutParams layoutParam =
            new LinearLayout.LayoutParams(
                    LayoutParams.FILL_PARENT,
                    LayoutParams.WRAP_CONTENT );

        this.addContentView(layout, layoutParam);
    }
}
```

(3) 按F11键在Android模拟器上调试应用程序。图3-21显示活动被创建了。



图 3-21

示例说明

本例中，首先注释掉setContentView()语句，这样就不从main.xml文件加载用户界面。然后，创建一个LayoutParams对象来指定可以被别的视图(接下来将创建)使用的布局参数：

```
        //---视图的参数---
        LayoutParams params =
            new LinearLayout.LayoutParams(
                LayoutParams.FILL_PARENT,
                LayoutParams.WRAP_CONTENT);
```

还创建一个LinearLayout对象来包含活动中所有的视图：

```
//---创建一个布局---
LinearLayout layout = new LinearLayout(this);
layout.setOrientation(LinearLayout.VERTICAL);
```

下一步，创建一个TextView视图和一个Button视图：

```
//---创建一个文本视图---
TextView tv = new TextView(this);
tv.setText("This is a TextView");
tv.setLayoutParams(params);

//---创建一个按钮---
Button btn = new Button(this);
btn.setText("This is a Button");
btn.setLayoutParams(params);
```

然后将它们添加到LinearLayout对象中：

```
//---添加文本视图---
layout.addView(tv);

//---添加按钮---
layout.addView(btn);
```

另外，创建一个被LinearLayout对象使用的LayoutParams对象：

```
//---为布局创建一个布局参数---
LinearLayout.LayoutParams layoutParam =
    new LinearLayout.LayoutParams(
            LayoutParams.FILL_PARENT,
            LayoutParams.WRAP_CONTENT );
```

最后，将LinearLayout对象添加到活动中：

```
this.addContentView(layout, layoutParam);
```

我们可以看出，使用代码创建用户界面是一个非常费力的工作。因此，只有在必要时才使用代码来动态生成用户界面。

3.5　侦听用户界面通知

用户和用户界面在两个层面上进行交互：活动层面和视图层面。在活动层面，Activity类暴露了可以被重写的方法。一些常见的可以在活动中被重写的方法如下所示：

- onKeyDown——当一个键被按下并且没有被活动中的任何视图处理时调用
- onKeyUp——当一个键被释放并且没有被活动中的任何视图处理时调用
- onMenuItemSelected——当用户选择了面板的菜单时调用(见第5章)
- onMenuOpened——当用户打开了面板的菜单时调用(见第5章)

3.5.1 重写活动中定义的方法

为了理解活动是如何与用户交互的，让我们从重写活动的基类中定义的一些方法说起，看一看当用户和活动交互时，这些方法是如何被处理的。

试一试　重写活动的方法

UIActivity.zip代码文件可以在Wrox.com上下载

(1) 使用Eclipse，创建一个新的Android项目，并命名为UIActivity。

(2) 在main.xml文件中添加下列粗体显示的语句：

```
<? xml version="1.0" encoding="utf-8"? >
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TextView
        android:layout_width="214dp"
        android:layout_height="wrap_content"
        android:text="Your Name"
        />
    <EditText
        android:id="@+id/txt1"
        android:layout_width="214dp"
        android:layout_height="wrap_content"
        />
    <Button
        android:id="@+id/btn1"
        android:layout_width="106dp"
        android:layout_height="wrap_content"
        android:text="OK"
        />
    <Button
        android:id="@+id/btn2"
        android:layout_width="106dp"
        android:layout_height="wrap_content"
        android:text="Cancel"
        />
</LinearLayout>
```

(3) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.UIActivity;

import android.app.Activity;
import android.os.Bundle;
```

```
import android.view.KeyEvent;
import android.widget.Toast;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        switch (keyCode)
        {
            case KeyEvent.KEYCODE_DPAD_CENTER:
                Toast.makeText(getBaseContext(),
                        "Center was clicked",
                        Toast.LENGTH_LONG).show();
                break;
            case KeyEvent.KEYCODE_DPAD_LEFT:
                Toast.makeText(getBaseContext(),
                        "Left arrow was clicked",
                        Toast.LENGTH_LONG).show();
                break;
            case KeyEvent.KEYCODE_DPAD_RIGHT:
                Toast.makeText(getBaseContext(),
                        "Right arrow was clicked",
                        Toast.LENGTH_LONG).show();
                break;
            case KeyEvent.KEYCODE_DPAD_UP:
                Toast.makeText(getBaseContext(),
                        "Up arrow was clicked",
                        Toast.LENGTH_LONG).show();

                break;
            case KeyEvent.KEYCODE_DPAD_DOWN:
                Toast.makeText(getBaseContext(),
                        "Down arrow was clicked",
                        Toast.LENGTH_LONG).show();
                break;
        }
        return false;
    }
}
```

(4) 按F11键在Android模拟器上调试应用程序。

(5) 当活动加载后，输入一些文本，如图3-22左边所示。下一步，单击方向键盘上的下箭头键。观察屏幕上显示的消息，如图3-22右边黑色区域所示。

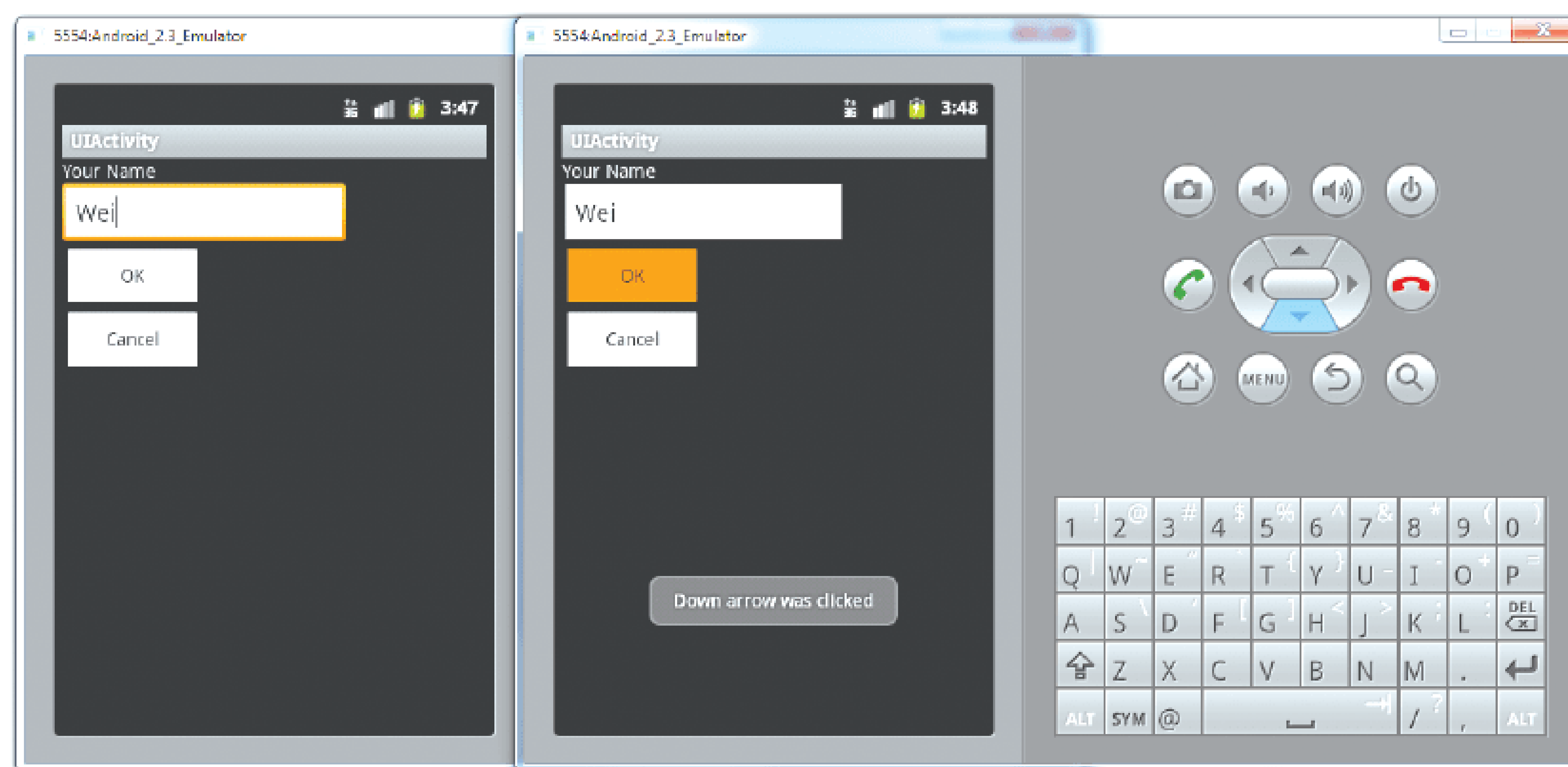


图 3-22

示例说明

当加载活动时，光标将在EditText视图中闪烁，因为它获得了焦点。

在MainActivity类中，按如下所示重写Activity基类的onKeyDown()方法：

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event)
{
    switch (keyCode)
    {
        case KeyEvent.KEYCODE_DPAD_CENTER:
            //...
            break;
        case KeyEvent.KEYCODE_DPAD_LEFT:
            //...
            break;
        case KeyEvent.KEYCODE_DPAD_RIGHT:
            //...
            break;
        case KeyEvent.KEYCODE_DPAD_UP:
            //...
            break;
        case KeyEvent.KEYCODE_DPAD_DOWN:
            //...
            break;
    }
    return false;
}
```

在Android中，当您按下设备上的任意一个键时，当前获得焦点的视图将试图处理生成的事件。本例中，当EditText获得焦点并且您按下了一个键时，EditText视图将处理这一事件并在视图中将您刚刚输入的字符显示出来。然而，如果您按了上或下箭头键，EditText视图不会对此作处理，相反会将这个事件传递给活动。在这种情况下，将调用onKeyDown()方法。本例中，您检查按下的键并显示一条消息表明该键被按下了。可以看到，现在焦点转移到了下一个视图

上，即OK按钮。

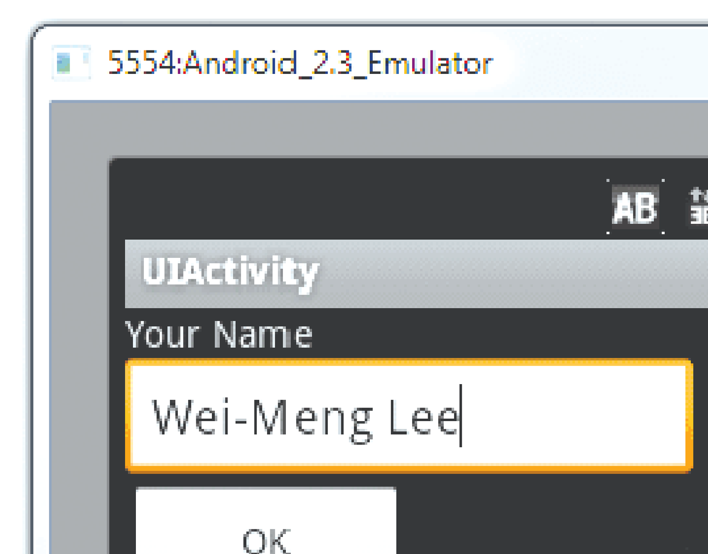


图 3-23

有趣的是，如果EditText视图内已经有了一些文本，并且光标位于文本末尾(如图3-23所示)，那么单击左箭头键不会触发onKeyDown()事件，而只是将光标向左移动一个字符。这是由于EditText视图已经处理了这一事件。如果按下的是右箭头键，将调用onKeyDown()方法(因为现在EditText视图将不会处理这一事件)。这同样适用于光标位于EditText视图开始的情况：单击左箭头将触发onKeyDown()事件，而单击右箭头将只是将光标向右移动一个字符。

当OK按钮获得焦点时，按下方向键盘上的中间按钮。可以看到Center was clicked消息没有显示出来。这是由于Button视图本身处理了单击事件。因此，onKeyDown()方法没有捕获这一事件。不过，如果此时没有任何视图获得焦点的话(可以通过单击屏幕背景实现)，那么按下中间键将显示Center was clicked消息(如图3-24所示)。

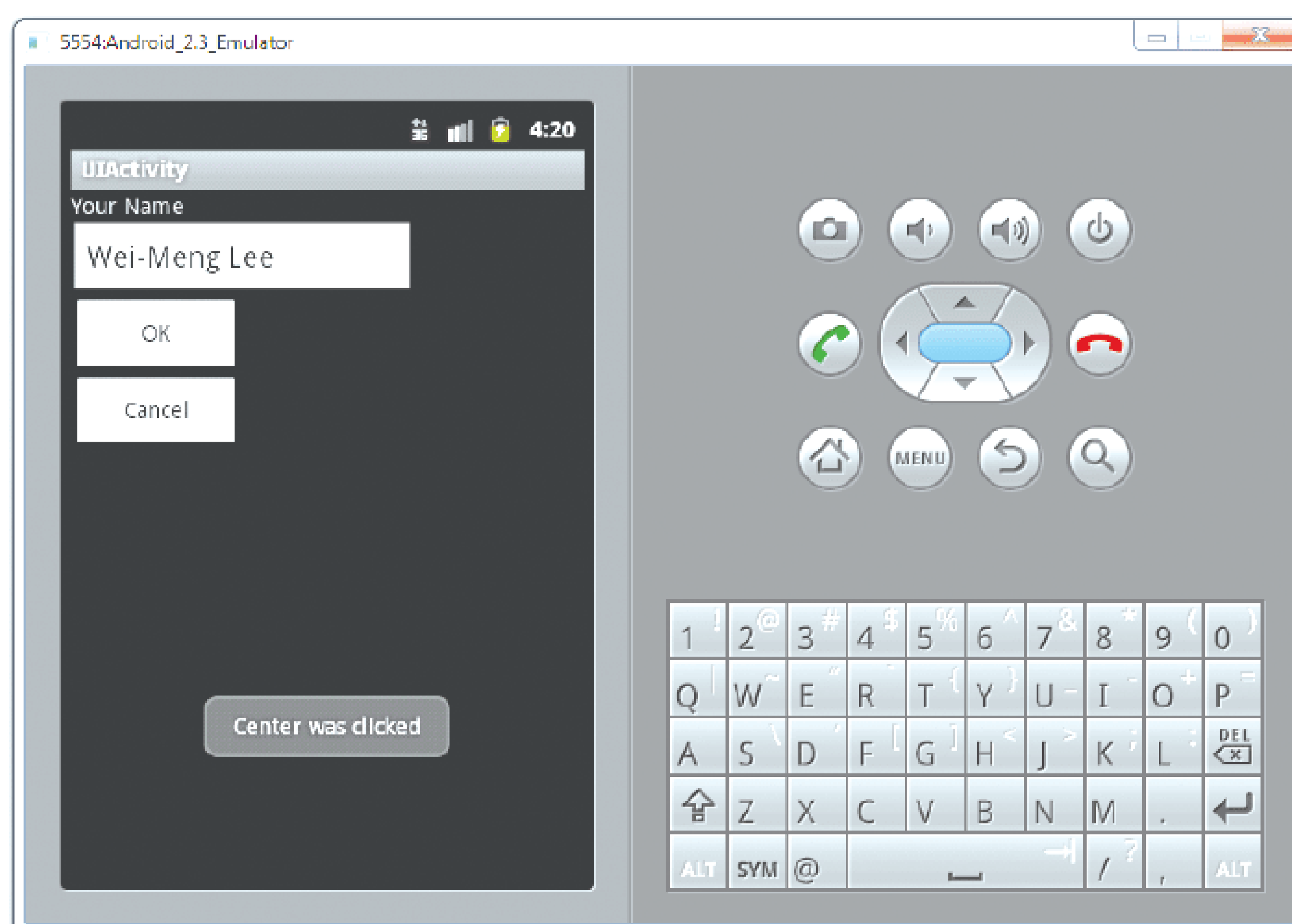


图 3-24

注意，onKeyDown()方法返回一个boolean类型的结果。当想告诉系统您已经处理完此事件并且系统不要再作进一步的处理时，应当返回true。例如，考虑下列当每一个键匹配后返回true时的情况：

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event)
{
    switch (keyCode)
    {
        case KeyEvent.KEYCODE_DPAD_CENTER:
            Toast.makeText(getBaseContext(),
                    "Center was clicked",
                    Toast.LENGTH_LONG).show();
            return true;
    case KeyEvent.KEYCODE_DPAD_LEFT:
            Toast.makeText(getBaseContext(),
                    "Left arrow was clicked",
```

```
                        Toast.LENGTH_LONG).show();
                return true;
            case KeyEvent.KEYCODE_DPAD_RIGHT:
                Toast.makeText(getBaseContext(),
                        "Right arrow was clicked",
                        Toast.LENGTH_LONG).show();
                return true;
            case KeyEvent.KEYCODE_DPAD_UP:
                Toast.makeText(getBaseContext(),
                        "Up arrow was clicked",
                        Toast.LENGTH_LONG).show();
                return true;
            case KeyEvent.KEYCODE_DPAD_DOWN:
                Toast.makeText(getBaseContext(),
                        "Down arrow was clicked",
                        Toast.LENGTH_LONG).show();
                return true;
        }
        return false;
    }
```

如果测试一下，就会发现现在不能使用箭头键在视图之间导航。

3.5.2 为视图注册事件

当用户和视图交互时，视图可以触发事件。例如，当用户触碰一个Button视图时，您需要响应这个事件以便能够采取适当的动作。要做到这一点，需要为视图显式地注册事件。

使用上一节讨论的同一个例子，我们知道那个活动有两个Button视图；因此，可以使用一个匿名类注册按钮单击事件，如下所示：

```
package net.learn2develop.UIActivity;

import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.widget.Toast;

import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //---两个按钮被连接到同一个事件处理程序---
```

```
        Button btn1 = (Button)findViewById(R.id.btn1);
        btn1.setOnClickListener(btnListener);

        Button btn2 = (Button)findViewById(R.id.btn2);
        btn2.setOnClickListener(btnListener);
    }

    //---创建一个匿名类作为按钮单击的侦听器---
    private OnClickListener btnListener = new OnClickListener()
    {
        public void onClick(View v)
        {
            Toast.makeText(getBaseContext(),
                    ((Button) v).getText() + " was clicked",
                    Toast.LENGTH_LONG).show();
        }
    };

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        switch (keyCode)
        {
                //...
                //...
        }
        return false;
    }
}
```

现在，无论您按下的是OK按钮还是Cancel按钮，都会显示出相应的消息(如图3-25所示)，这证明该事件被正确地连接起来了。

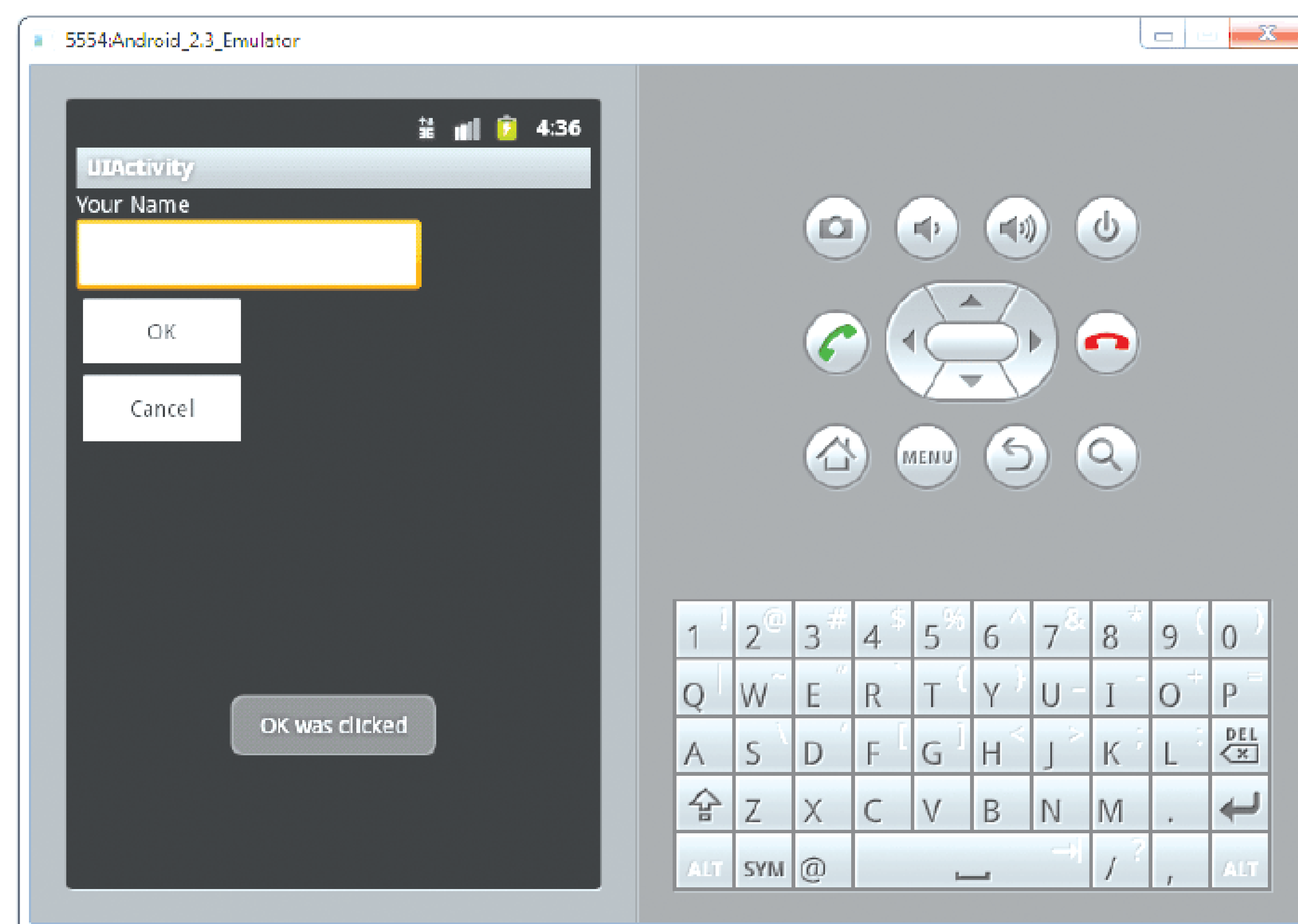


图 3-25

除了为事件处理程序定义一个匿名类外，还可以定义一个匿名内部类来处理事件。下面的示例显示了如何为EditText视图处理onFoucusChange()事件。

```
import android.widget.EditText;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //---两个按钮被连接到同一个事件处理程序---
        Button btn1 = (Button)findViewById(R.id.btn1);
        btn1.setOnClickListener(btnListener);

        Button btn2 = (Button)findViewById(R.id.btn2);
        btn2.setOnClickListener(btnListener);

        EditText txt1 = (EditText)findViewById(R.id.txt1);

        //---创建一个匿名内部类作为获得焦点事件的侦听器---
        txt1.setOnFocusChangeListener(new View.OnFocusChangeListener()
        {
            @Override
            public void onFocusChange(View v, boolean hasFocus) {
                Toast.makeText(getBaseContext(),
                    ((EditText) v).getId() + " has focus - " + hasFocus,
                    Toast.LENGTH_LONG).show();
            }
        });
}
```

当EditText视图收到焦点时，屏幕上将输出一条消息，如图3-26所示。

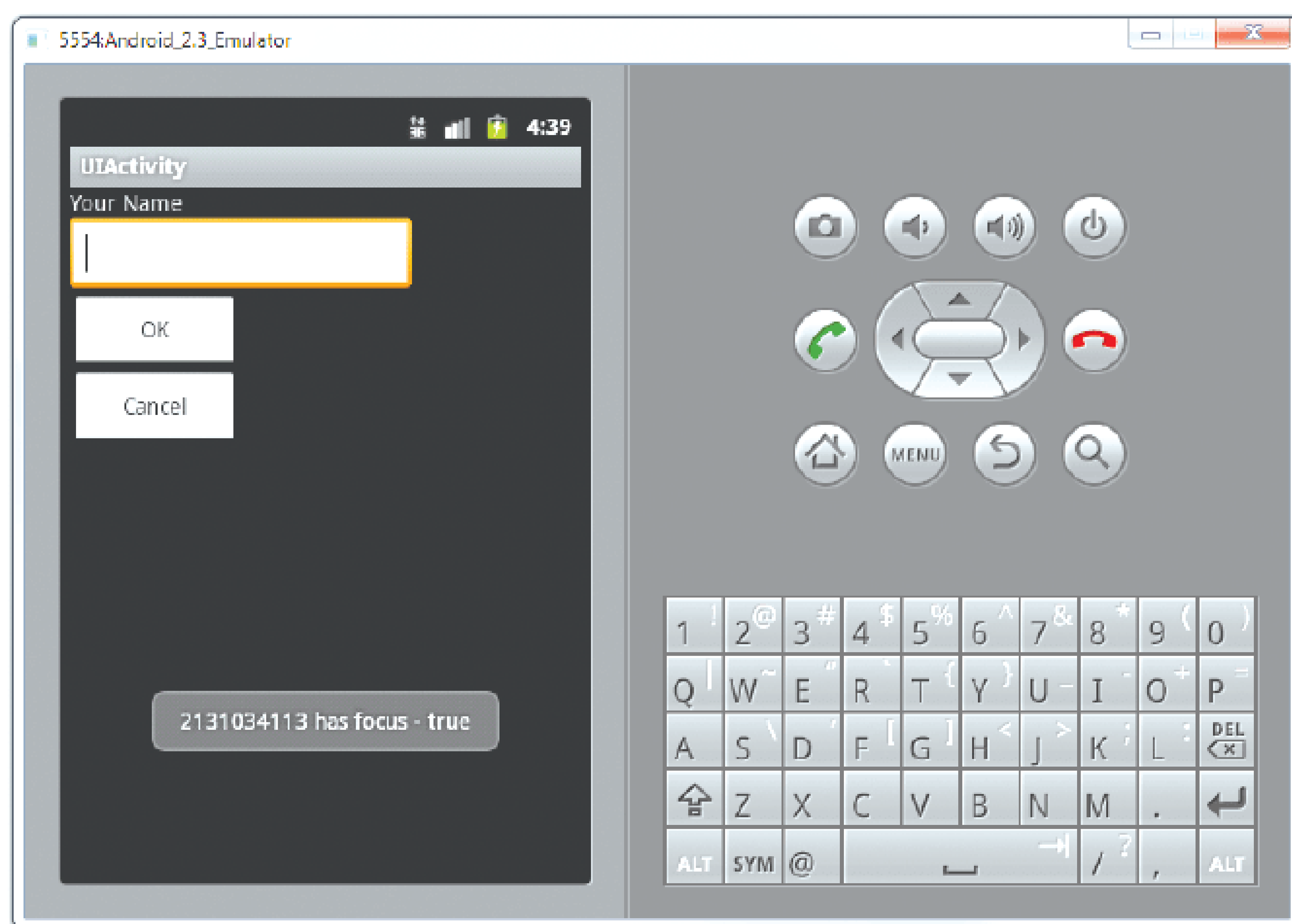


图 3-26

3.6　本章小结

本章学习了如何在Android中创建用户界面，还学习了可以用来在Android用户界面中定位视图的不同布局。由于Android设备支持多种屏幕方向，所以需要特别注意这一点，以确保您的用户界面能够适应屏幕方向的改变。

练　习

1. dp单位和px单位有何不同？应该用哪一个来指定视图的尺寸？
2. 为什么不建议使用AbsoluteLayout？
3. onPause()事件和onSaveInstanceState()事件有何区别？
4. 列举3个可以重写来保存活动状态的事件。

练习答案参见附录C。

本章主要内容

主　　题	关 键 概 念
LinearLayout	以单行或单列的形式排列视图
AbsoluteLayout	可用于指定其子元素的确切位置
TableLayout	以行和列的形式组织视图
RelativeLayout	可用于指定子视图相对于彼此之间是如何定位的
FrameLayout	一个在屏幕上可以用来显示单个视图的占位符
ScrollView	一种特殊类型的FrameLayout，因为它可以使用户滚动显示一个占据的空间大于物理显示的视图列表
度量单位	使用dp指定视图尺寸，使用sp指定字体大小
适应方向变化的两种方法	锚定与调整大小和重新定位
为不同方向使用不同的XML文件	纵向用户界面使用layout文件夹，横向用户界面使用layout-land文件夹
保持活动状态的3种方法	使用onPause()事件 使用onSaveInstanceState()事件 使用onRetainNonConfigurationInstance()事件
获得当前设备的尺寸	使用WindowManager类的getDefaultDisplay()方法
限制活动的方向	使用setRequestOrientation()方法或者AndroidManifest.xml文件中的android:screenOrientation属性

第4章 使用视图设计用户界面

本章将介绍以下内容

- 如何使用Android中的基本视图设计用户界面
- 如何使用选取器视图显示项列表
- 如何使用列表视图显示项列表

第3章中学习了可以用来在一个活动中定位视图的不同布局，还学习了可以用来适应不同屏幕分辨率和尺寸的相关技术。本章中，您将了解到可以用来为应用程序设计用户界面的各种视图。

特别地，将学习到以下视图组：

- 基本视图——常用的视图，如TextView、EditText和Button视图
- 选取器视图——可以使用户从一个列表中进行选择的视图，如TimePicker和DatePicker视图
- 列表视图——显示长的项列表的视图，如ListView和SpinnerView视图

随后的章节将涵盖本章未讨论到的其他视图，如日期和时间的选取器视图以及用于显示图形的其他视图等。

4.1 基本视图

首先，我们探究一些可以用于设计Android应用程序的用户界面的基本视图：

- TextView
- EditText
- Button
- ImageButton
- CheckBox
- ToggleButton
- RadioButton
- RadioGroup

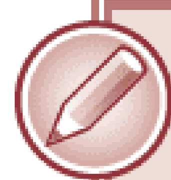
这些基本视图可以用来显示文本信息以及执行一些基本的选择。下面的章节将详细研究所有这些视图。

4.1.1　TextView视图

当创建一个新的Android项目时，Eclipse将创建一个包含<TextView>元素的main.xml文件(位于res/layout文件夹下)：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

TextView视图用来向用户显示文本。这是最基本的视图，在开发Android应用程序时会频繁用到。如果想让用户可以编辑显示的文本，则应该使用TextView的子类——EditText，4.1.2节将讨论它。



注意： 在某些平台上，TextView常被称为标签视图。其唯一的目的就是在屏幕上显示文本。

4.1.2　Button、ImageButton、EditText、CheckBox、ToggleButton、RadioButton和RadioGroup视图

除了最经常用到的TextView视图之外，还有其他一些您将频繁使用到的基本控件：Button、ImageButton、EditText、CheckBox、ToggleButton、RadioButton和RadioGroup。

- Button——表示一个按钮的小部件。
- ImageButton——与Button视图类似，另外还显示一个图像。
- EditText——TextView视图的子类，另外还允许用户编辑其文本内容。
- CheckBox——具有两个状态的特殊按钮类型：选中或未选中。
- RadioGroup和RadioButton——RadioButton有两个状态：选中或未选中。一旦一个RadioButton被选中，它就不能被取消选中了。RadioGroup用来把一个或多个RadioButton视图组合在一起，从而在该RadioGroup中只允许一个RadioButton被选中。
- ToggleButton——用一个灯光指示器来显示选中/未选中状态。

下面的“试一试”揭示了这些视图工作原理的细节。

试一试　使用基本视图

BasicViews1.zip代码文件可以在Wrox.com上下载

(1) 使用Eclipse，按图4-1所示创建并命名一个Android项目。

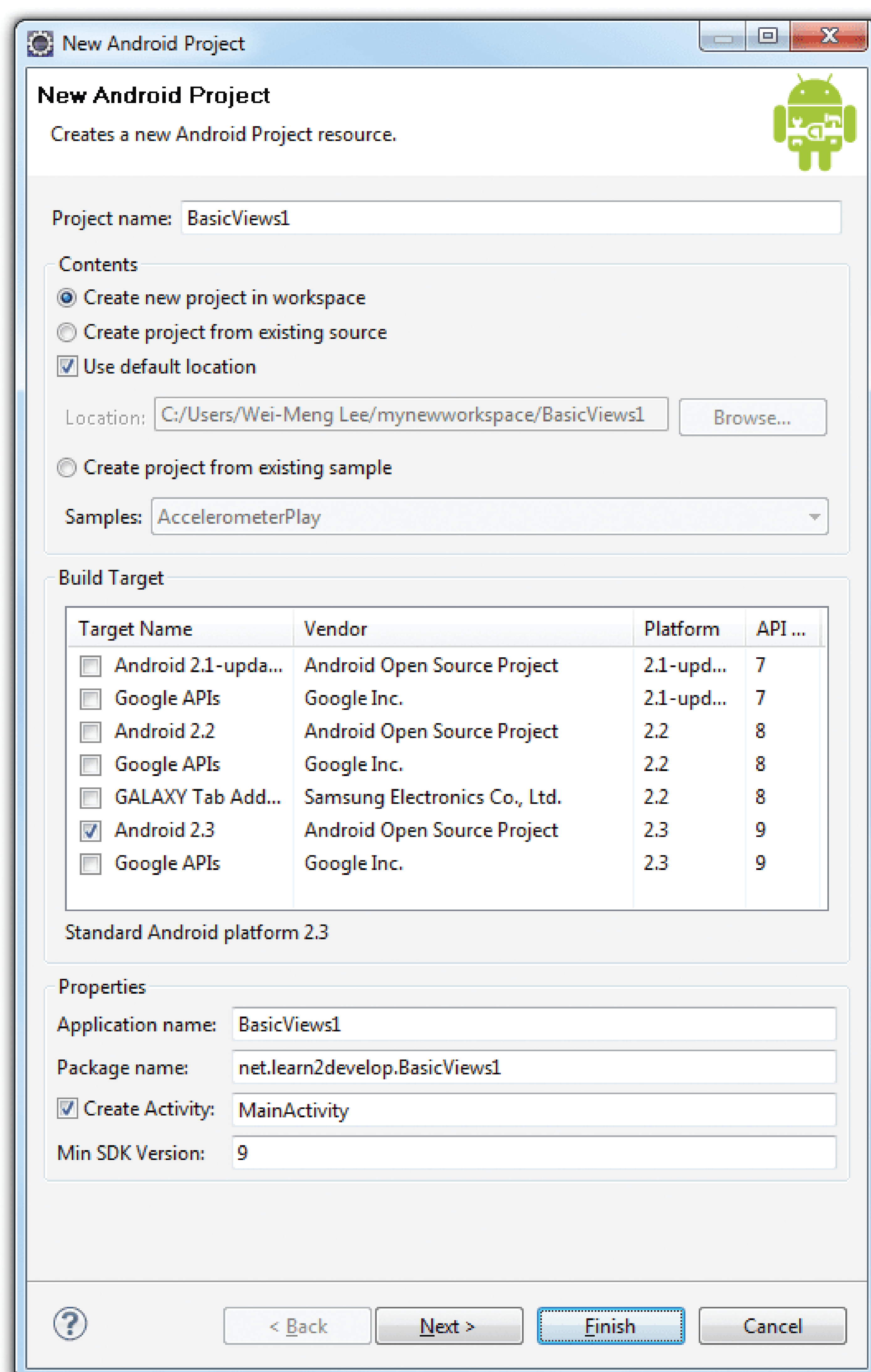
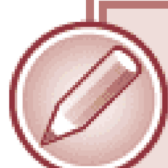


图 4-1



注意：本书后续创建的项目中的各项字段将采用以下的值：

- Application Name：<*project name*>
- Package Name：net.learn2develop.<*project name*>
- Create Activity：MainActivity
- Min SDK Version：9

(2) 在位于res/layout文件夹下的main.xml文件中添加下列粗体显示的元素：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <Button android:id="@+id/btnSave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Save" />
```

```
    <Button android:id="@+id/btnOpen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Open" />

    <ImageButton android:id="@+id/btnImg1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/icon" />

    <EditText android:id="@+id/txtName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <CheckBox android:id="@+id/chkAutosave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Autosave" />

    <CheckBox android:id="@+id/star"
        style="?android:attr/starStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <RadioGroup android:id="@+id/rdbGp1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >
        <RadioButton android:id="@+id/rdb1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Option 1" />
        <RadioButton android:id="@+id/rdb2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Option 2" />
    </RadioGroup>

    <ToggleButton android:id="@+id/toggle1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

(3) 要观察视图的效果，可在Eclipse中选择项目名称并按F11键进行调试。图4-2展示了在Android模拟器中显示的不同视图。



图 4-2

(4) 单击不同的视图并注意它们在外观和感觉上的变化。图4-3展示了视图的以下改变：

- 第1个CheckBox视图(Autosave)被选中。
- 第2个CheckBox视图(星形)被选中。
- 第2个RadioButton(Option 2)被选中。
- ToggleButton被打开。

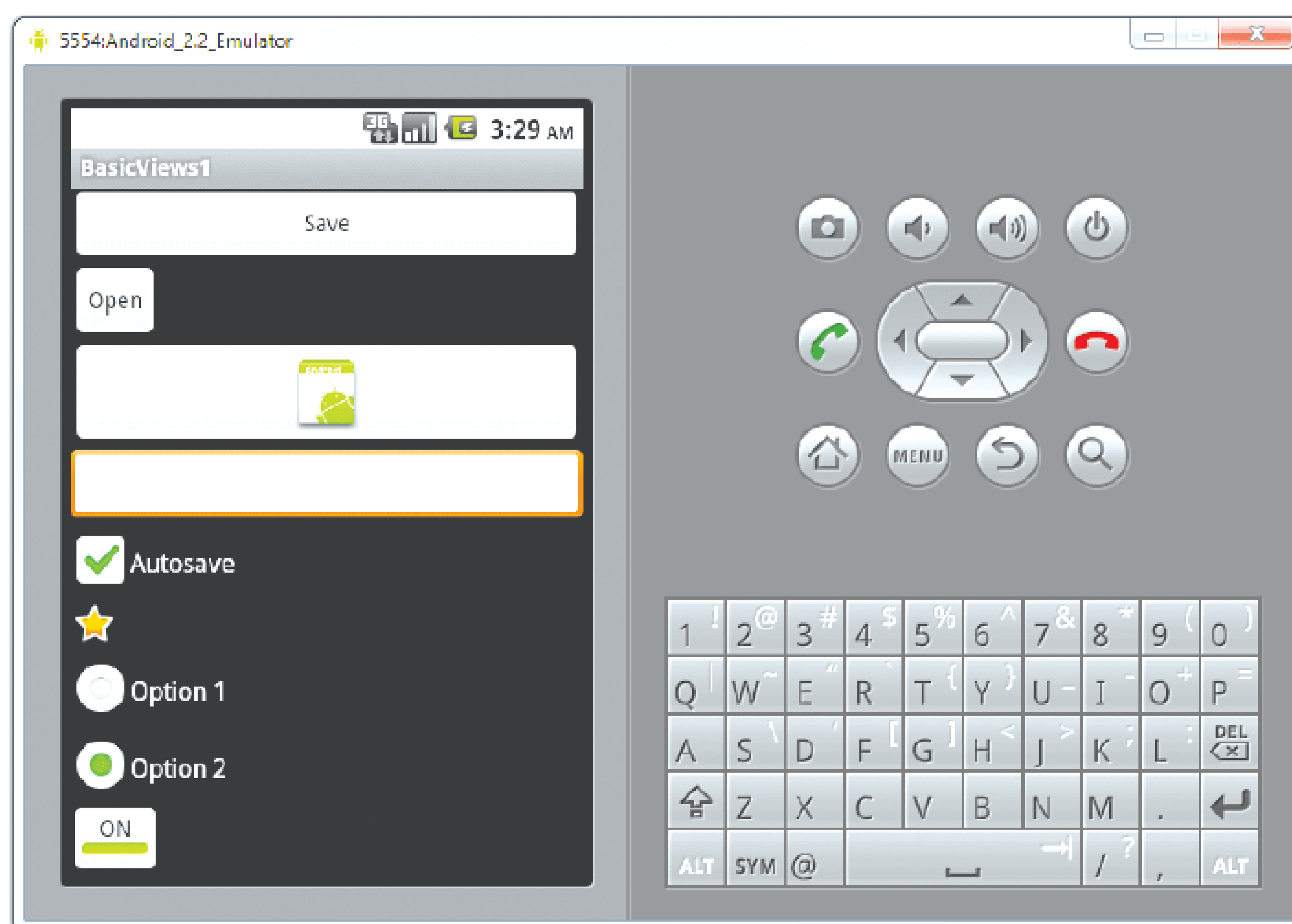


图 4-3

示例说明

到目前为止，所有的视图都是相对简单的——使用<LinearLayout>元素将它们一一列出，因此当在活动中显示时，它们堆叠在彼此之上。

对于第1个Button，layout_width属性被设置为fill_parent，因此其宽度将占据整个屏幕的宽度：

```
<Button android:id="@+id/btnSave"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Save" />
```

对于第2个Button，layout_width属性被设置为wrap_content，因此其宽度将是其所包含内容的宽度——具体来说，就是显示的文本(也就是Open)：

```
<Button android:id="@+id/btnOpen"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Open" />
```

ImageButton显示了一个带有图像的按钮。图像通过src属性设置。本例中，使用曾用作应用程序图标的图像：

```
<ImageButton android:id="@+id/btnImg1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:src="@drawable/icon" />
```

EditText视图显示了一个矩形区域，用户可以向其中输入一些文本。layout_height属性被设置为wrap_content，这样，如果用户输入一个长的文本串，EditText的高度将随着内容自动调整(如图4-4所示)。

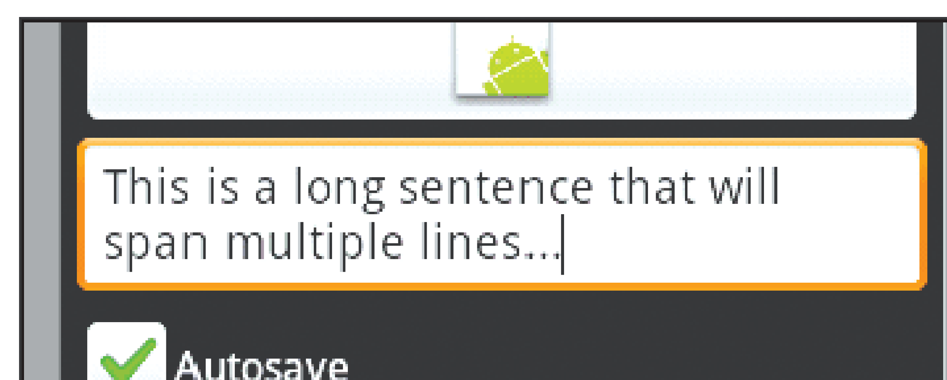


图 4-4

```
<EditText android:id="@+id/txtName"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

CheckBox显示了一个用户可以通过轻点鼠标进行选中或取消选中的复选框：

```
<CheckBox android:id="@+id/chkAutosave"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Autosave" />
```

如果您不喜欢CheckBox的默认外观，可以对其应用一个样式属性，使其显示为其他图像，如星形：

```
<CheckBox android:id="@+id/star"
    style="?android:attr/starStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

样式属性的值的格式如下所示：

```
?[package:][type:]name
```

RadioGroup包含了两个RadioButton。这一点很重要，因为单选按钮通常用来表示多个选项以便用户选择。当选择了RadioGroup中的一个RadioButton时，其他所有RadioButton就自动取消选择：

```
<RadioGroup android:id="@+id/rdbGp1"
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >
        <RadioButton android:id="@+id/rdb1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Option 1" />
        <RadioButton android:id="@+id/rdb2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Option 2" />
    </RadioGroup>
```

注意，RadioButton是垂直排列的，一个位于另一个之上。如果想要水平排列，需要把orientation属性改为horizontal。还需要确保RadioButton的layout_width属性被设置为wrap_content：

```
    <RadioGroup android:id="@+id/rdbGp1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <RadioButton android:id="@+id/rdb1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Option 1" />
        <RadioButton android:id="@+id/rdb2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Option 2" />
</RadioGroup>
```

图4-5显示了水平排列的RadioButton。

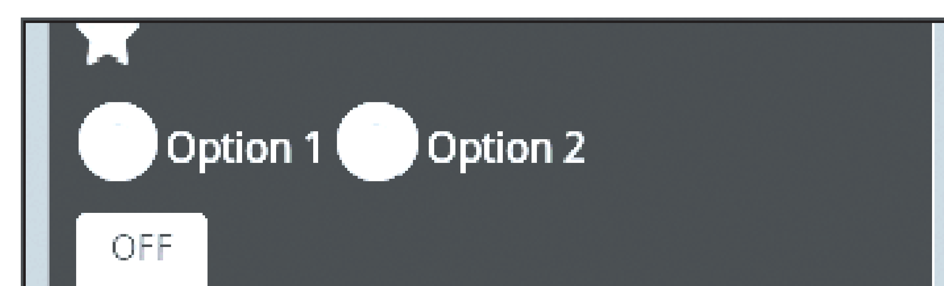


图 4-5

ToggleButton显示了一个矩形按钮，用户可以通过单击它来实现开和关的切换：

```
    <ToggleButton android:id="@+id/toggle1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

这个例子中，始终保持一致的一件事情是每个视图都有一个设置为特定值的id属性，如Button视图中所示：

```
    <Button android:id="@+id/btnSave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Save" />
```

id属性是视图的标识符，因此可以在以后使用View.findViewById()或Activity.findViewById()方法来检索它。

现在，您已经了解了一个活动的多种视图的外观，下面的“试一试”将教您如何以编程方式控制它们。

试一试 处理视图事件

(1) 使用前面的“试一试”所创建的同一个项目，修改MainActivity.java文件，添加下列粗体显示的语句：

```
package net.learn2develop.BasicViews1;

import android.app.Activity;
import android.os.Bundle;

import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;
import android.widget.ToggleButton;
import android.widget.RadioGroup.OnCheckedChangeListener;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //---Button视图---
        Button btnOpen = (Button) findViewById(R.id.btnOpen);
        btnOpen.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                DisplayToast("You have clicked the Open button");
            }
        });

        //---Button视图---
        Button btnSave = (Button) findViewById(R.id.btnSave);
        btnSave.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v) {
                DisplayToast("You have clicked the Save button");
            }
        });

        //---CheckBox---
        CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);
        checkBox.setOnClickListener(new View.OnClickListener()
```

```
            {
                public void onClick(View v) {
                    if (((CheckBox)v).isChecked())
                        DisplayToast("CheckBox is checked");
                    else
                        DisplayToast("CheckBox is unchecked");
                }
            });

            //---RadioButton---
            RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdbGp1);
            radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener()
            {
                public void onCheckedChanged(RadioGroup group, int checkedId) {
                    RadioButton rb1 = (RadioButton) findViewById(R.id.rdb1);
                    if (rb1.isChecked()) {
                        DisplayToast("Option 1 checked!");
                    } else {
                        DisplayToast("Option 2 checked!");
                    }
                }
            });

            //---ToggleButton---
            ToggleButton toggleButton =
                (ToggleButton) findViewById(R.id.toggle1);
            toggleButton.setOnClickListener(new View.OnClickListener()
            {
                public void onClick(View v) {
                   if (((ToggleButton)v).isChecked())
                        DisplayToast("Toggle button is On");
                   else
                       DisplayToast("Toggle button is Off");
                }
            });
        }

        private void DisplayToast(String msg)
        {
             Toast.makeText(getBaseContext(), msg,
                     Toast.LENGTH_SHORT).show();
        }
    }
```

(2) 按F11键在Android模拟器中调试项目。

(3) 单击不同的视图，观察在Toast窗口中显示的消息。

示例说明

为了处理每一个视图所触发的事件，首先需要以编程方式定位在onCreate()事件中所创建的

视图。做法是使用Acitivity.findViewById()方法，传入该视图的ID。

```
//---Button视图---
Button btnOpen = (Button) findViewById(R.id.btnOpen);
```

setOnClickListener()方法注册一个回调函数，以便在后面视图被单击时调用。

```
btnOpen.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        DisplayToast("You have clicked the Open button");
    }
});
```

当单击视图时，将调用onClick()方法。

对于CheckBox，为了确定其状态，必须把onClick()方法的参数类型转换成一个CheckBox，然后检查它的isChecked()方法来确定其是否被选中：

```
//---CheckBox---
CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);
checkBox.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View v) {
        if (((CheckBox)v).isChecked())
            DisplayToast("CheckBox is checked");
        else
            DisplayToast("CheckBox is unchecked");
    }
});
```

对于RadioButton，需要使用RadioGroup的setOnCheckedChangeListener()方法注册一个回调函数，以便在该组中被选中的RadioButton发生变化时调用：

```
//---RadioButton---
RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdbGp1);
radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener()
{
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        RadioButton rb1 = (RadioButton) findViewById(R.id.rdb1);
        if (rb1.isChecked()) {
            DisplayToast("Option 1 checked!");
        } else {
            DisplayToast("Option 2 checked!");
        }
    }
});
```

当选中一个RadioButton时，将触发onCheckedChanged()方法。在这一过程中，找到那些单个的RadioButton，然后调用它们的isChecked()方法来确定是哪个RadioButton被选中。或者，onCheckedChanged()方法包含第2个参数，其中包含被选定RadioButton的唯一标识符。

4.1.3 ProgressBar视图

ProgressBar视图提供了一些正在进行的任务的视觉反馈，如当您在后台执行一个任务时。例如，您可能正从Web上下载一些数据并需要更新用户的下载状态。在这种情况下，使用ProgressBar视图来完成这一任务是一个不错的选择。

试一试　使用ProgressBar视图

BasicViews2.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个名为BasicViews2的Android项目。

(2) 修改位于res/layout文件夹下的main.xml文件，添加下列粗体显示的代码：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

<ProgressBar android:id="@+id/progressbar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

</LinearLayout>
```

(3) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.BasicViews2;

import android.app.Activity;
import android.os.Bundle;

import android.os.Handler;
import android.widget.ProgressBar;

public class MainActivity extends Activity {

    private static int progress;
    private ProgressBar progressBar;
    private int progressStatus = 0;
    private Handler handler = new Handler();

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        progress = 0;
```

```
        progressBar = (ProgressBar) findViewById(R.id.progressbar);

        //---在后台线程中做一些工作---
        new Thread(new Runnable()
        {
            public void run()
            {
                //---这里做一些工作---
                while (progressStatus < 10)
                {
                    progressStatus = doSomeWork();
                }

                //---隐藏进度条---
                handler.post(new Runnable()
                {
                    public void run()
                    {
                        //---0 - VISIBLE; 4 - INVISIBLE; 8 - GONE---
                        progressBar.setVisibility(8);
                    }
                });
            }

            //---这里做一些费时的工作---
            private int doSomeWork()
            {
                try {
                    //---模拟做一些工作---
                    Thread.sleep(500);
                } catch (InterruptedException e)
                {
                    e.printStackTrace();
                }
                return ++progress;
            }
        }).start();
    }
}
```

(4) 按F11键在Android模拟器中调试项目。图4-6显示了ProgressBar的动画。大约5秒钟后，它将消失。



图 4-6

示例说明

ProgressBar视图的默认模式是不确定的——也就是说，它显示一个循环的动画。这种模式对于完成时间没有明确指示的任务是非常有用的，例如

当您向一个Web服务发送一些数据并等待服务器的响应时。如果只是把<ProgressBar>元素放入main.xml文件中，它会不断地显示一个旋转的图标。当后台任务已经完成时，需要您来使它停止旋转。

已在Java文件中添加的代码显示了如何分配一个后台线程来模拟执行一些长时间运行的任务。要做到这一点，可以配合使用Thread类和一个Runnable对象。run()方法启动线程的执行，在这种情况下调用doSomeWork()方法来模拟做一些工作。当模拟工作完成后(大约5秒种之后)，使用Handler对象给线程发送一条消息来取消ProgressBar：

```
//---在后台线程中做一些工作---
new Thread(new Runnable()
{
    public void run()
    {
        //---这里做一些工作---
        while (progressStatus < 10)
        {
            progressStatus = doSomeWork();
        }

        //---隐藏进度条---
        handler.post(new Runnable()
        {
            public void run()
            {
                //---0 - VISIBLE; 4 - INVISIBLE; 8 - GONE---
                progressBar.setVisibility(8);
            }
        });
    }

    //---这里做一些费时的工作---
    private int doSomeWork()
    {
        try {
            //---模拟做一些工作---
            Thread.sleep(500);
        } catch (InterruptedException e)
        {
            e.printStackTrace();
        }
        return ++progress;
    }
}).start();
```

当任务完成时，通过设置ProgressBar的Visibility属性为GONE(值8)来隐藏它。INVISIBLE和GONE常量的区别在于INVISIBLE常量只是隐藏ProgressBar(ProgressBar所占区域仍旧在活动中占据空间)。GONE常量则从活动中移除ProgressBar视图，不再占据任何空间。

下面的“试一试”演示了如何改变ProgressBar的外观。

试一试　定制ProgressBar视图

(1) 使用前面的“试一试”中所创建的同一个项目，按如下所示修改main.xml文件：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

<ProgressBar android:id="@+id/progressbar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    style="?android:attr/progressBarStyleHorizontal" />

</LinearLayout>
```

(2) 修改MainActivity.java文件，添加下列粗体显示的语句：

```
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        progress = 0;
        progressBar = (ProgressBar) findViewById(R.id.progressbar);
        progressBar.setMax(200);

        //---在后台线程中做一些工作---
        new Thread(new Runnable()
        {
            public void run()
            {
                //---这里做一些工作---
                while (progressStatus < 100)
                {
                    progressStatus = doSomeWork();

                    //---更新进度条---
                    handler.post(new Runnable()
                    {
                        public void run() {
                            progressBar.setProgress(progressStatus);
                        }
                    });
                }

                //---隐藏进度条---
                handler.post(new Runnable()
```

```
                {
                    public void run()
                    {
                        //---0 - VISIBLE; 4 - INVISIBLE; 8 - GONE---
                        progressBar.setVisibility(8);
                    }
                });
            }

            //---这里做一些费时的工作---
            private int doSomeWork()
            {
                try {
                    //---模拟做一些工作---
                    Thread.sleep(50);
                } catch (InterruptedException e)
                {
                    e.printStackTrace();
                }
                return ++progress;
            }
        }).start();
    }
```

(3) 按F11键在Android模拟器中调试项目。

(4) 图4-7展示了正显示进度的ProgressBar。当进度达到50%时，ProgressBar消失。



图 4-7

示例说明

为了使ProgressBar水平显示，只要设置其style属性为?android:attr/progressBarStyleHorizontal:

```
<ProgressBar android:id="@+id/progressbar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    style="?android:attr/progressBarStyleHorizontal" />
```

为了显示进度，调用setProgress()方法，传入一个表示进度的整数：

```
//---更新进度条---
handler.post(new Runnable()
{
    public void run() {
        progressBar.setProgress(progressStatus);
    }
});
```

在本例中，设置了ProgressBar的范围为0～200(通过setMax()方法)。因此，ProgressBar将在中途停止并消失(由于仅仅在progressStatus小于100时才持续调用doSomeWork()方法)。为了确保ProgressBar只有当进度达到100%时才消失，可以设置最大值为100，或者修改while循环为当progressStatus达到200时就停止，如下所示：

```
//---这里做一些工作---
while (progressStatus < 200)
```

4.1.4　AutoCompleteTextView视图

AutoCompleteTextView是一种与EditText类似的视图(实际上，它是EditText的子类)，另外它还在用户输入时自动显示完成建议的列表。下面的“试一试”展示了如何利用AutoCompleteTextView来自动协助用户完成文本输入：

试一试　使用AutoCompleteTextView

BasicViews3.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个名为BasicViews3的Android项目。

(2) 按如下粗体显示内容修改位于res/layout文件夹下的main.xml文件：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Name of President" />
```

```
<AutoCompleteTextView android:id="@+id/txtCountries"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

</LinearLayout>
```

(3) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.BasicViews3;

import android.app.Activity;
import android.os.Bundle;

import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;

public class MainActivity extends Activity {
    String[] presidents = {
        "Dwight D. Eisenhower",
        "John F. Kennedy",
        "Lyndon B. Johnson",
        "Richard Nixon",
        "Gerald Ford",
        "Jimmy Carter",
        "Ronald Reagan",
        "George H. W. Bush",
        "Bill Clinton",
        "George W. Bush",
        "Barack Obama"
    };

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_dropdown_item_1line, presidents);

        AutoCompleteTextView textView = (AutoCompleteTextView)
            findViewById(R.id.txtCountries);

        textView.setThreshold(3);
        textView.setAdapter(adapter);
    }
}
```

(4) 按F11键在Android模拟器中调试应用程序。如图4-8所示，在向AutoCompleteTextView中输入时，会随之显示一个匹配名字的列表。

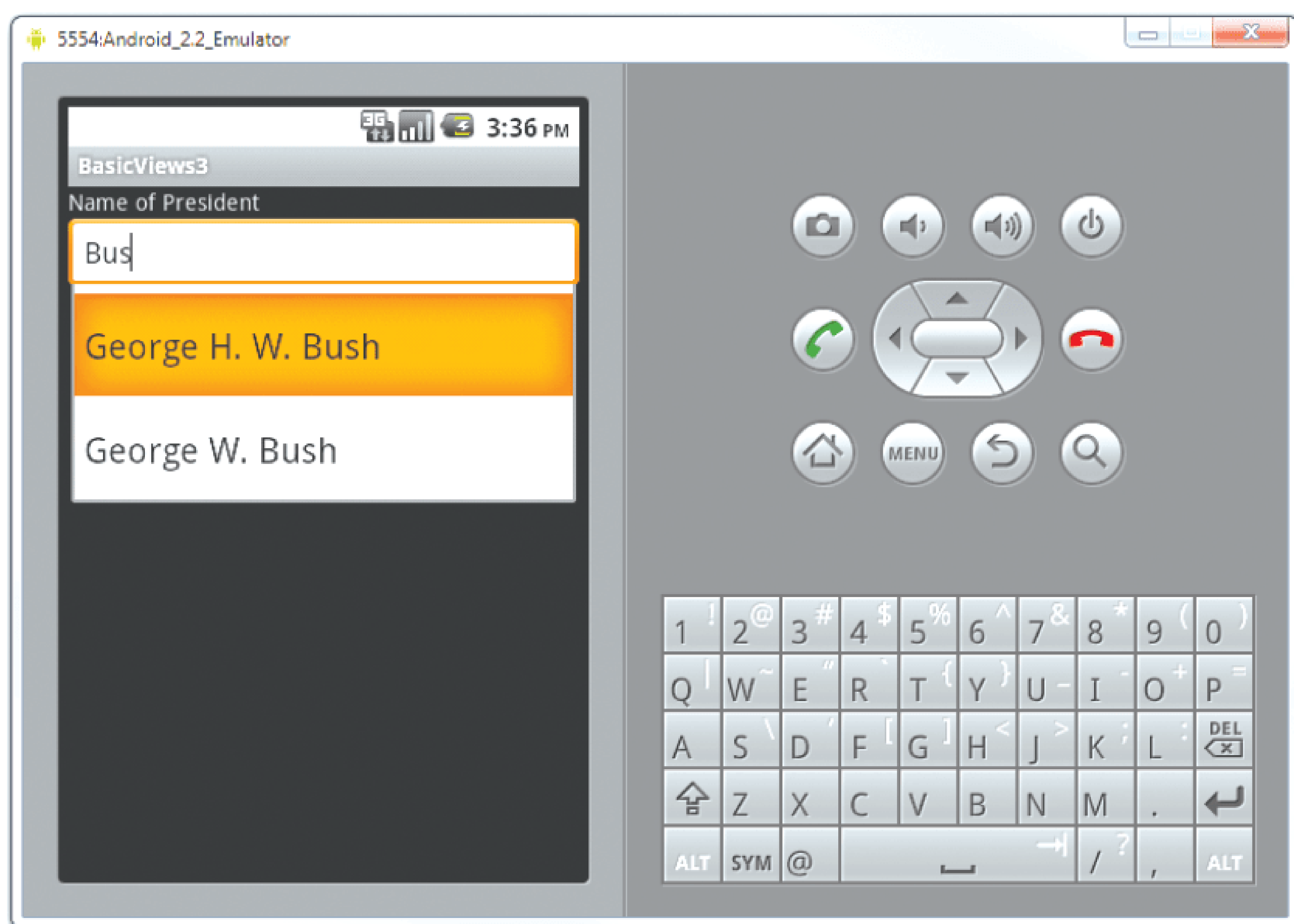


图 4-8

示例说明

在MainActivity类中，首先创建了一个包含一组总统名字的String数组：

```
String[] presidents = {
    "Dwight D. Eisenhower",
    "John F. Kennedy",
    "Lyndon B. Johnson",
    "Richard Nixon",
    "Gerald Ford",
    "Jimmy Carter",
    "Ronald Reagan",
    "George H. W. Bush",
    "Bill Clinton",
    "George W. Bush",
    "Barack Obama"
};
```

ArrayAdapter对象管理将由AutoCompleteTextView显示的字符串数组。在先前的例子中，将AutoCompleteTextView设置为以simple_dropdown_item_1line模式显示：

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_dropdown_item_1line, presidents);
```

setThreshold()方法设置建议以下拉菜单形式出现前用户必须输入的最少字符个数：

```
textView.setThreshold(3);
```

为AutoCompleteTextView显示的建议列表从ArrayAdapter对象获得：

```
textView.setAdapter(adapter);
```

4.2 选取器视图

选择日期和时间是您在一个移动应用程序中需要执行的常见任务之一。Android通过TimePicker和DatePicker视图来支持这一功能。下面的小节将阐述如何在活动中使用这些视图。

4.2.1 TimePicker视图

TimePicker视图可以使用户按24小时或AM/PM模式选择一天中的某个时间。下面的“试一试”展示了如何使用这一视图。

试一试　使用TimePicker视图

BasicViews4.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个名为BasicViews4的Android项目。

(2) 修改位于res/layout文件夹下的main.xml文件，添加下列粗体显示的行：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

<TimePicker android:id="@+id/timePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<Button android:id="@+id/btnSet"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="I am all set!" />

</LinearLayout>
```

(3) 按F11键在Android模拟器中调试应用程序。图4-9显示了运用中的TimePicker。除了单击加(+)和减(−)按钮外，还可以使用设备上的数字键盘来修改小时和分钟，单击AM按钮在AM和PM之间切换。

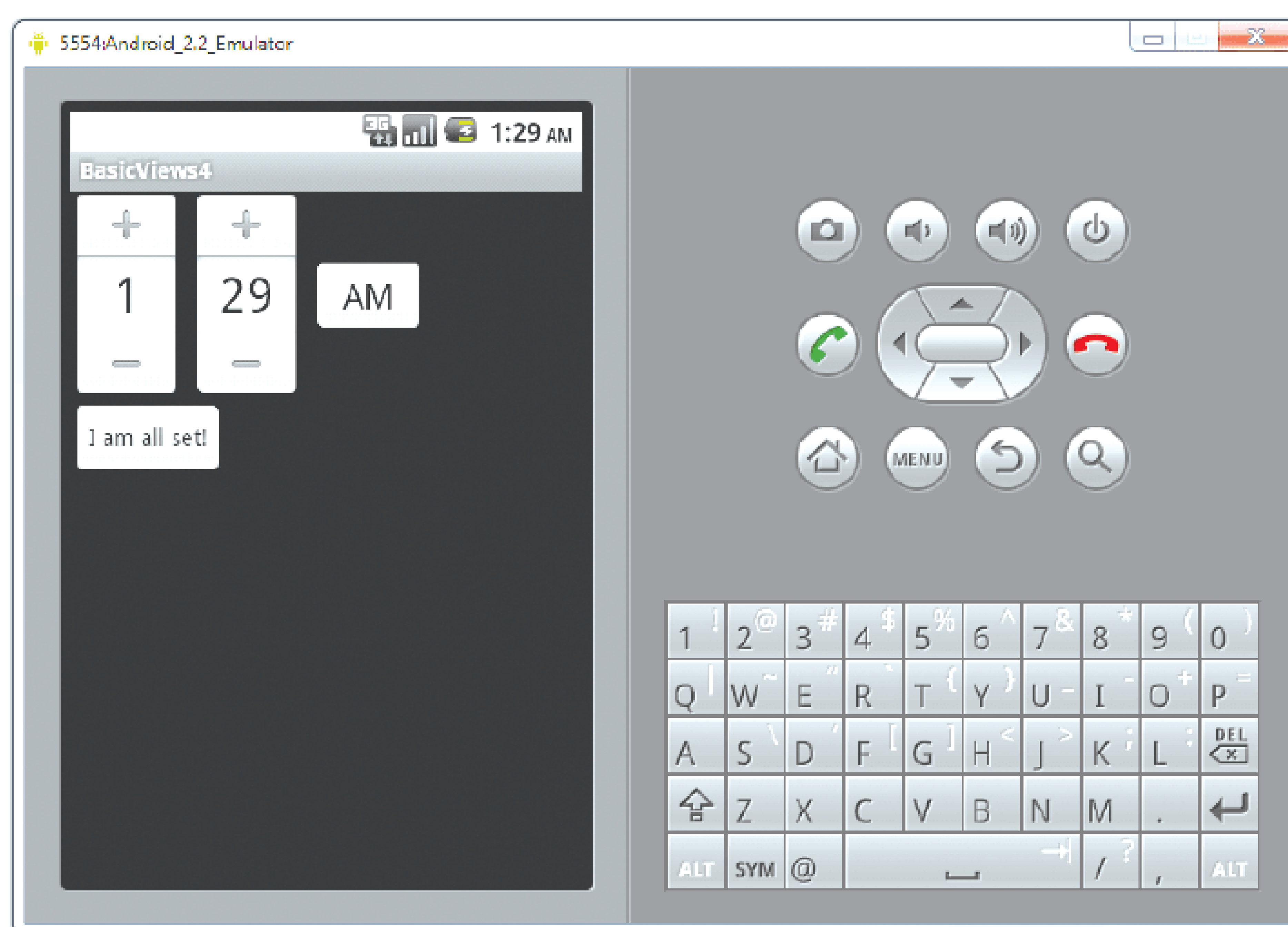


图 4-9

(4) 返回Eclipse，在MainAcitivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.BasicViews4;

import android.app.Activity;
import android.os.Bundle;

import android.view.View;
import android.widget.Button;
import android.widget.TimePicker;
import android.widget.Toast;

public class MainActivity extends Activity {
    TimePicker timePicker;

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        timePicker = (TimePicker) findViewById(R.id.timePicker);
        timePicker.setIs24HourView(true);

        //---Button视图---
        Button btnOpen = (Button) findViewById(R.id.btnSet);
        btnOpen.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Toast.makeText(getBaseContext(),
                        "Time selected:" +
                          timePicker.getCurrentHour() +
                        ":" + timePicker.getCurrentMinute(),
```

```
                        Toast.LENGTH_SHORT).show();
                }
            });
        }
    }
```

(5) 按F11键在Android模拟器上调试应用程序。这一次，TimePicker将以24小时格式显示。单击Button将显示您在TimePicker中设置好的时间(如图4-10所示)。

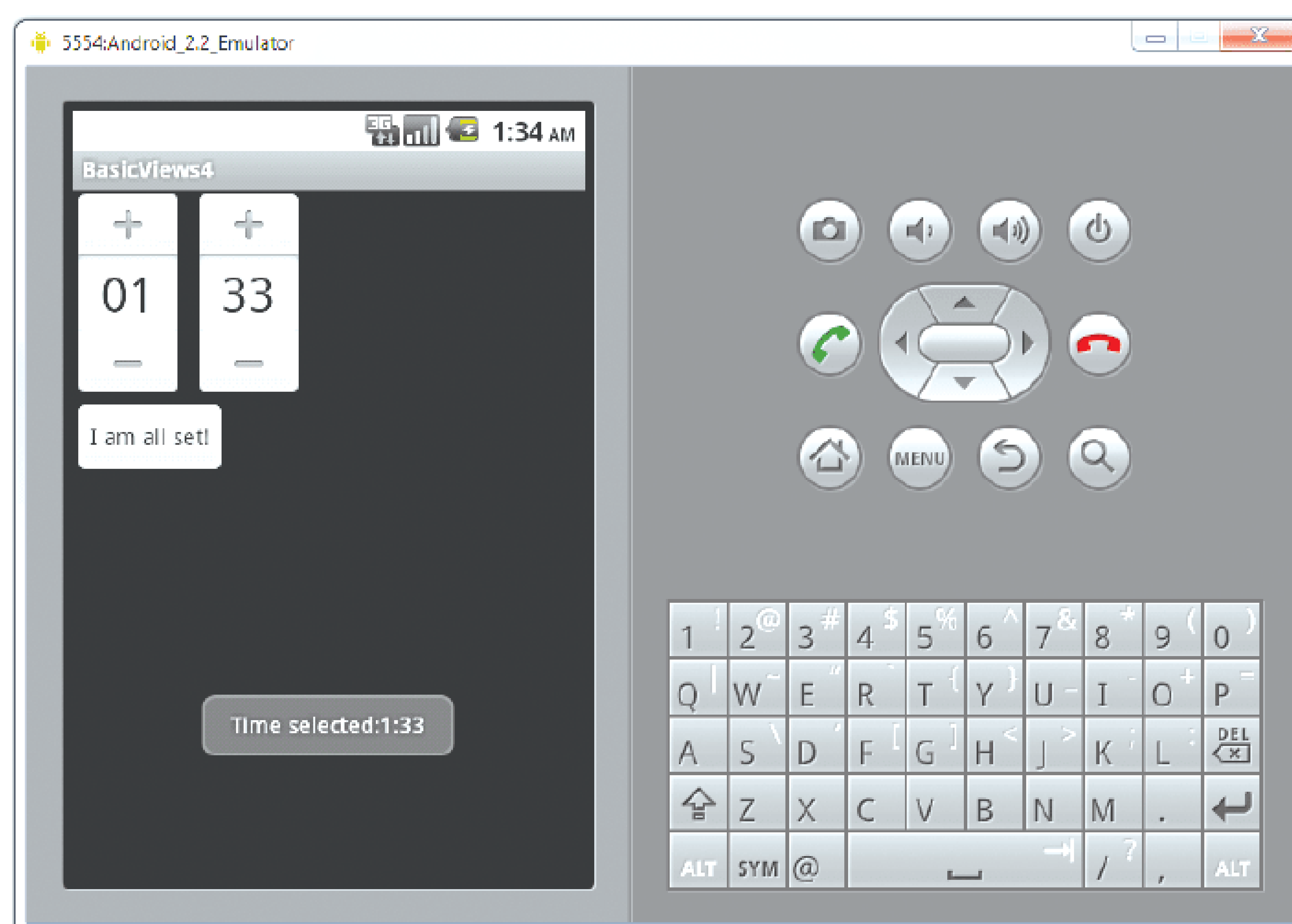


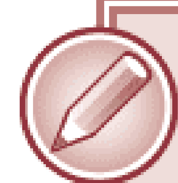
图 4-10

示例说明

TimePicker显示了一个可以让用户设置时间的标准用户界面。默认情况下，它以AM/PM格式显示时间。如果想要以24小时格式来显示，可以使用setIs24HourView()方法。

为了以编程方式获得用户设置的时间，可以使用getCurrentHour()和getCurrentMinute()方法：

```
Toast.makeText(getBaseContext(),
                "Time selected:" +
                 timePicker.getCurrentHour() +
                ":" + timePicker.getCurrentMinute(),
                Toast.LENGTH_SHORT).show();
```



注意： getCurrentHour()方法总是返回24小时格式的时间，也就是返回一个0～23之间的值。

在对话框窗口中显示TimePicker

虽然可以在一个活动中显示TimePicker，但更好的方法是在一个对话框窗口中显示它。这样一旦设置好时间，TimePicker就会消失，不再占据活动中的任何空间。下面的“试一试”展示了如何做到这一点。

试一试　使用对话框显示TimePicker视图

(1) 使用先前“试一试”中所创建的同一个项目，按如下所示修改MainActivity.java文件：

```
package net.learn2develop.BasicViews4;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TimePicker;
import android.widget.Toast;

import android.app.Dialog;
import android.app.TimePickerDialog;

public class MainActivity extends Activity {
    TimePicker timePicker;

    int hour, minute;
    static final int TIME_DIALOG_ID = 0;

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        showDialog(TIME_DIALOG_ID);

        timePicker = (TimePicker) findViewById(R.id.timePicker);
        timePicker.setIs24HourView(true);

        //---Button视图---
        Button btnOpen = (Button) findViewById(R.id.btnSet);
        btnOpen.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Toast.makeText(getBaseContext(),
                        "Time selected:" +
                           timePicker.getCurrentHour().toString() +
                        ":" + timePicker.getCurrentMinute().toString(),
                        Toast.LENGTH_SHORT).show();
                }
        });
    }

    @Override
    protected Dialog onCreateDialog(int id)
    {
        switch (id) {
```

```
                case TIME_DIALOG_ID:
                    return new TimePickerDialog(
                        this, mTimeSetListener, hour, minute, false);
            }
            return null;
        }

        private TimePickerDialog.OnTimeSetListener mTimeSetListener =
            new TimePickerDialog.OnTimeSetListener()
            {
                public void onTimeSet(
                TimePicker view, int hourOfDay, int minuteOfHour)
                {
                    hour = hourOfDay;
                    minute = minuteOfHour;
                    Toast.makeText(getBaseContext(),
                        "You have selected : " + hour + ":" + minute,
                        Toast.LENGTH_SHORT).show();
                }
            };
    }
```

(2) 按F11键在Android模拟器中调试应用程序。当活动被加载时，可以看到TimePicker显示在一个对话框窗口内(如图4-11所示)。设置一个时间，然后单击Set按钮，将看到Toast窗口显示了您刚刚设置好的时间。

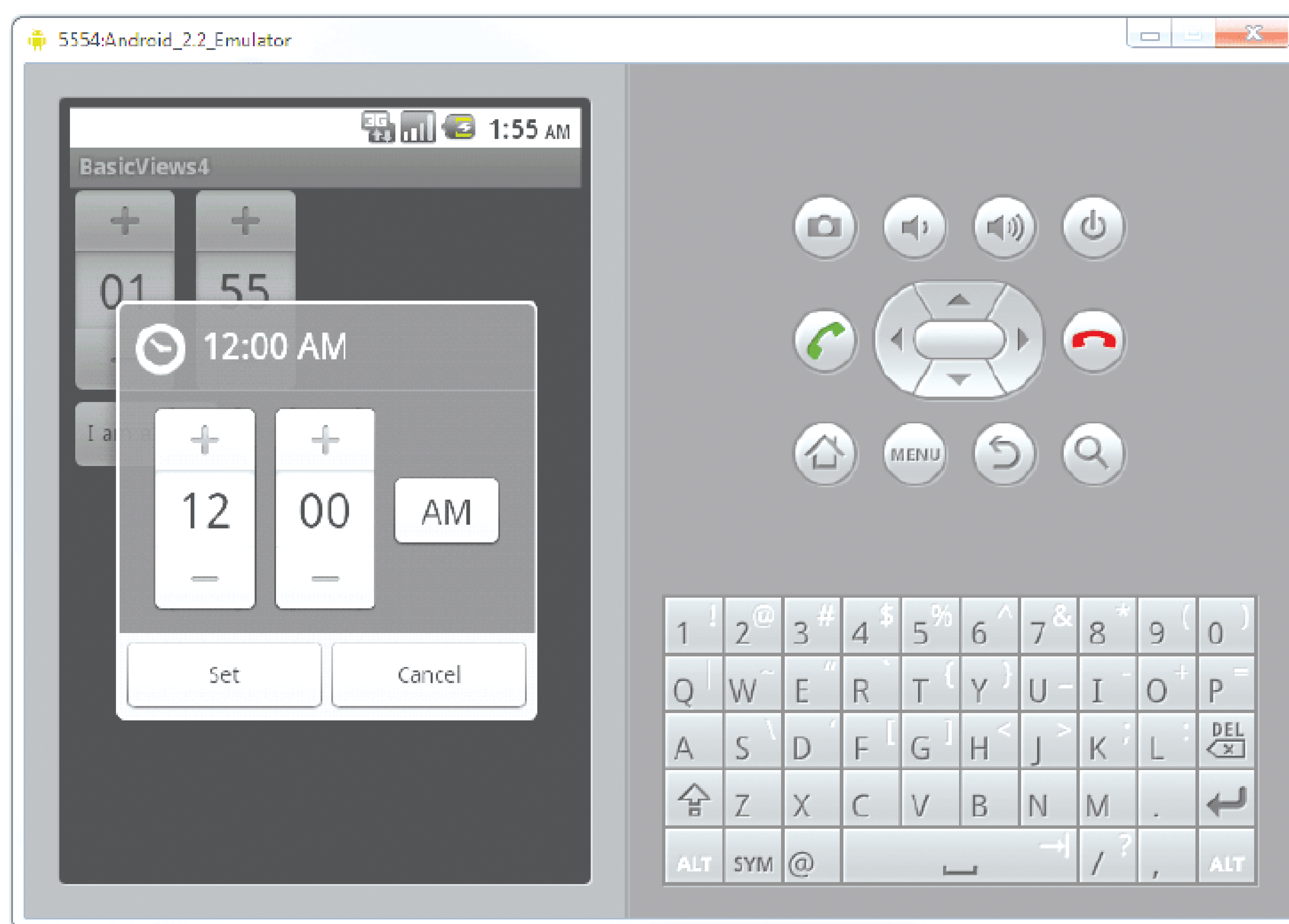


图 4-11

示例说明

为了显示一个对话框窗口，可以使用showDialog()方法，传入一个ID来标识对话框的源：

```
showDialog(TIME_DIALOG_ID);
```

当调用showDialog()方法时，onCreateDialog()方法将被调用：

```
@Override
protected Dialog onCreateDialog(int id)
{
    switch (id) {
        case TIME_DIALOG_ID:
            return new TimePickerDialog(
                this, mTimeSetListener, hour, minute, false);
    }
    return null;
}
```

这里，创建了一个TimePicker类的新实例，给它传递了当前上下文、回调函数、初始的小时和分钟，以及TimePicker是否以24小时格式显示的布尔常量。

当用户单击TimePicker对话框窗口中的Set按钮时，将调用onTimeSet()方法：

```
private TimePickerDialog.OnTimeSetListener mTimeSetListener =
    new TimePickerDialog.OnTimeSetListener()
    {
        public void onTimeSet(
        TimePicker view, int hourOfDay, int minuteOfHour)
        {
            hour = hourOfDay;
            minute = minuteOfHour;
            Toast.makeText(getBaseContext(),
                "You have selected : " + hour + ":" + minute,
                Toast.LENGTH_SHORT).show();
        }
    };
```

这里，onTiemSet()方法将包含用户分别通过hourOfDay和minuteOfHour参数设置的小时和分钟。

4.2.2 DatePicker视图

与TimePicker类似的另外一种视图就是DatePicker。利用DatePicker，可以使用户在活动中选择一个特定的日期。下面的“试一试”展示了如何使用DatePicker。

试一试 使用DatePicker视图

(1) 使用前述“试一试”中创建的同一个项目，按如下所示修改main.xml文件：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
```

```
        android:layout_height="fill_parent" >

    <DatePicker android:id="@+id/datePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TimePicker android:id="@+id/timePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <Button android:id="@+id/btnSet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I am all set!" />

    </LinearLayout>
```

(2) 按F11键在Android模拟器上调试应用程序。图4-12显示了DatePicker和TimePicker视图。

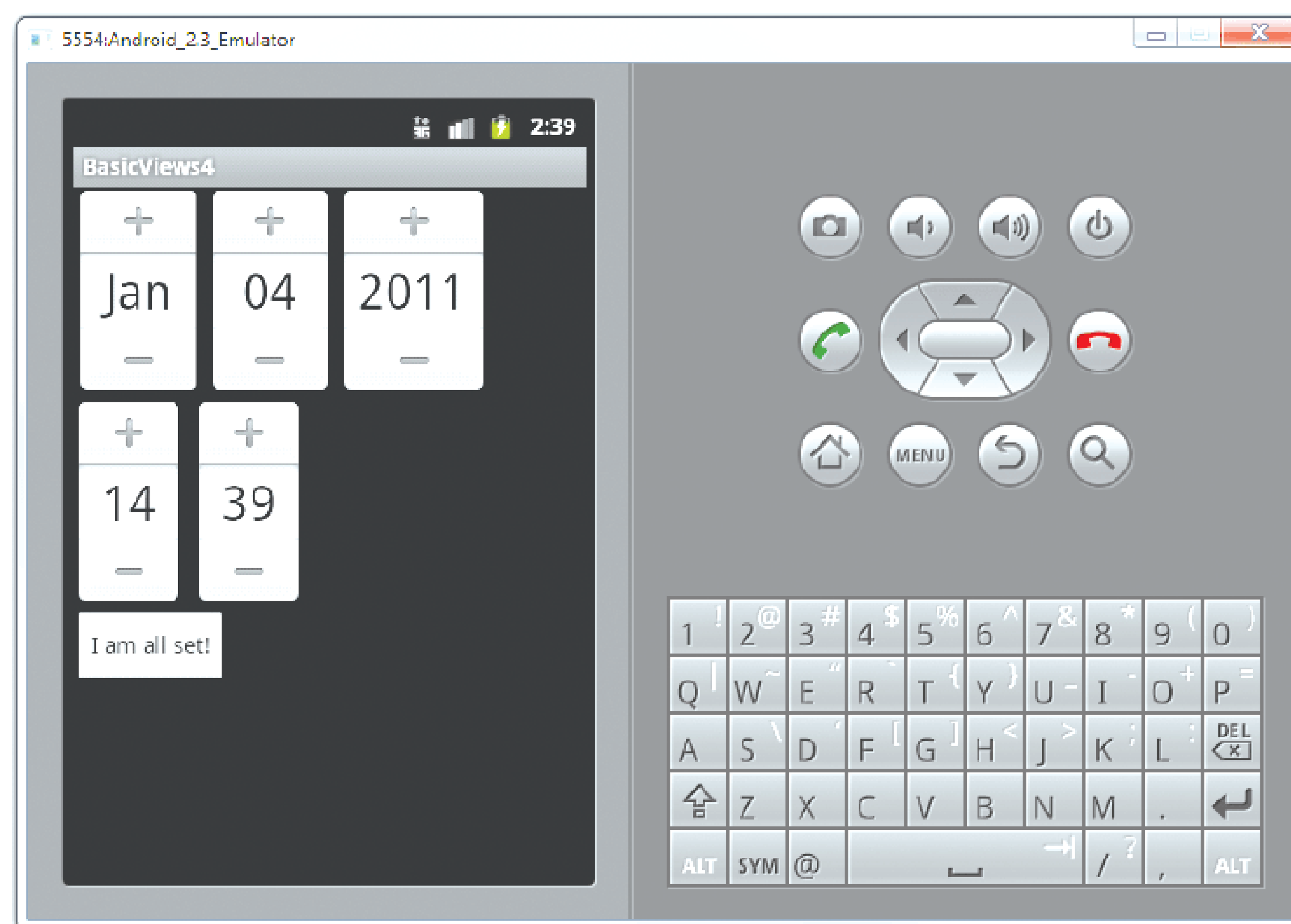


图 4-12

(3) 返回Eclipse，在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.BasicViews4;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import android.widget.Toast;

import android.app.Dialog;
import android.app.TimePickerDialog;
```

```
import android.widget.TimePicker;

import android.widget.DatePicker;

public class MainActivity extends Activity {
    TimePicker timePicker;
    DatePicker datePicker;

    int hour, minute;
    static final int TIME_DIALOG_ID = 0;

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //showDialog(TIME_DIALOG_ID);

        timePicker = (TimePicker) findViewById(R.id.timePicker);
        timePicker.setIs24HourView(true);

        datePicker = (DatePicker) findViewById(R.id.datePicker);

        //---Button视图---
        Button btnOpen = (Button) findViewById(R.id.btnSet);
        btnOpen.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Toast.makeText(getBaseContext(),
                        "Date selected:" + datePicker.getMonth() + 1 +
                        "/" + datePicker.getDayOfMonth() +
                        "/" + datePicker.getYear() + "\n" +
                        "Time selected:" + timePicker.getCurrentHour() +
                        ":" + timePicker.getCurrentMinute(),
                        Toast.LENGTH_SHORT).show();
            }
        });
    }

    @Override
    protected Dialog onCreateDialog(int id)
    {
        switch (id) {
            case TIME_DIALOG_ID:
                return new TimePickerDialog(
                    this, mTimeSetListener, hour, minute, false);
        }
        return null;
    }
```

```
        private TimePickerDialog.OnTimeSetListener mTimeSetListener =
            new TimePickerDialog.OnTimeSetListener()
            {
                public void onTimeSet(
                TimePicker view, int hourOfDay, int minuteOfHour)
                {
                    hour = hourOfDay;
                    minute = minuteOfHour;
                    Toast.makeText(getBaseContext(),
                        "You have selected : " + hour + ":" + minute,
                        Toast.LENGTH_SHORT).show();
                }
            };
    }
```

(4) 按F11键在Android模拟器上调试应用程序。一旦设置了日期，单击Button将显示设置的日期。

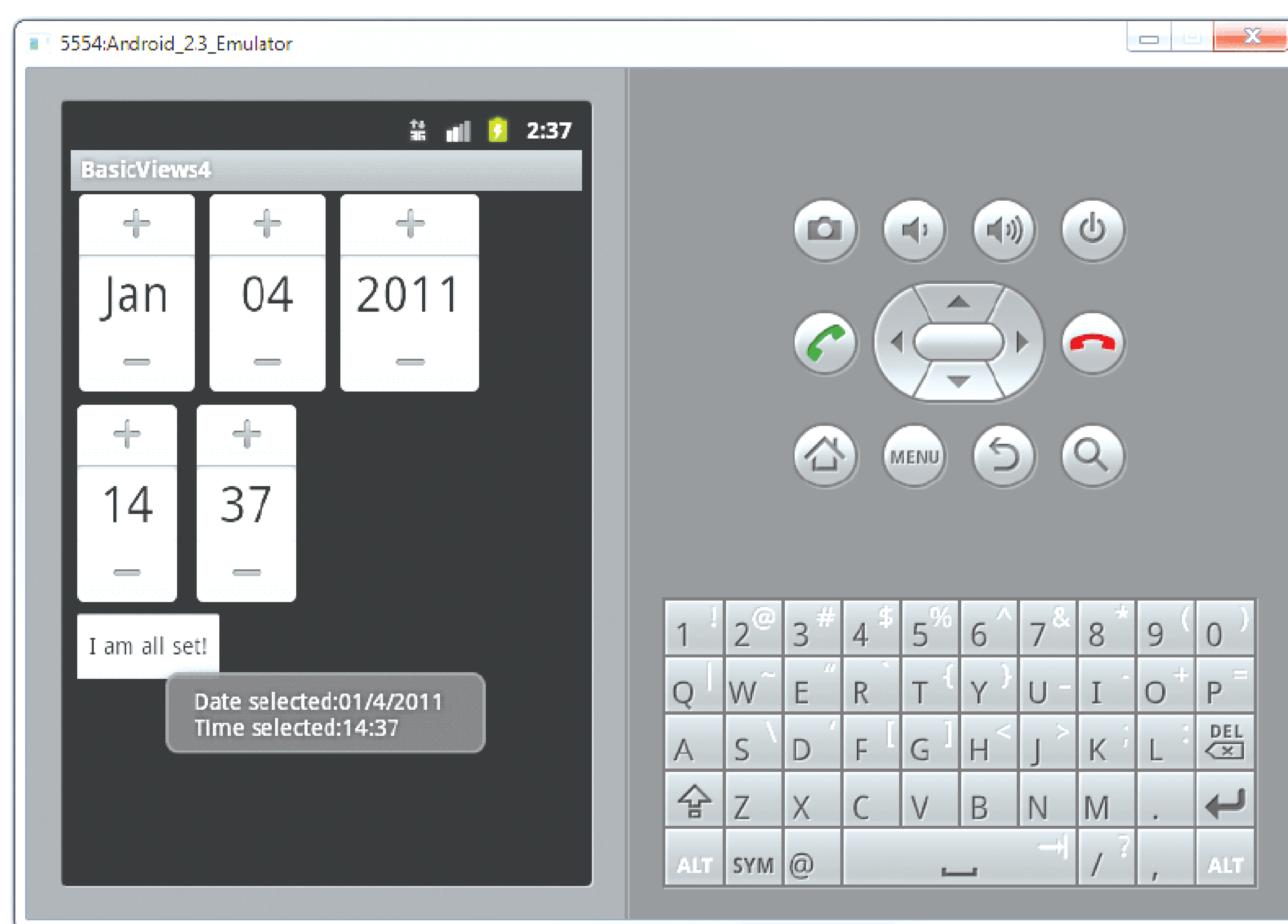


图 4-13

示例说明

与TimePicker类似，通过调用getMonth()、getDayOfMonth()和getYear()方法来分别获取月份、日子和年份：

```
"Date selected:" + datePicker.getMonth() + 1 +
"/" + datePicker.getDayOfMonth() +
"/" + datePicker.getYear() + "\n" +
```

注意，getMonth()方法返回0代表一月、返回1代表二月，依次类推。因此，需要将此方法返回的结果加1来获得月份数。

在对话框窗口中显示DatePicker视图

像TimePicker一样，也可以在对话框窗口中显示DatePicker。下面的“试一试”将教您如何做到这一点。

试一试　使用对话框显示DatePicker视图

(1) 使用前述“试一试”中创建的同一个项目，修改MainActivity.java文件，添加下列粗体显示的语句：

```
package net.learn2develop.BasicViews4;

import android.app.Activity;
import android.os.Bundle;

import android.view.View;
import android.widget.Button;

import android.widget.Toast;

import android.app.Dialog;
import android.app.TimePickerDialog;

import android.widget.TimePicker;
import android.widget.DatePicker;

import android.app.DatePickerDialog;
import java.util.Calendar;

public class MainActivity extends Activity {
    TimePicker timePicker;
    DatePicker datePicker;

    int hour, minute;
    int yr, month, day;

    static final int TIME_DIALOG_ID = 0;
    static final int DATE_DIALOG_ID = 1;

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //showDialog(TIME_DIALOG_ID);

        //---获取当前日期---
        Calendar today = Calendar.getInstance();
```

```
        yr = today.get(Calendar.YEAR);
        month = today.get(Calendar.MONTH);
        day = today.get(Calendar.DAY_OF_MONTH);
        showDialog(DATE_DIALOG_ID);

        timePicker = (TimePicker) findViewById(R.id.timePicker);
        timePicker.setIs24HourView(true);

        datePicker = (DatePicker) findViewById(R.id.datePicker);

        //---Button视图---
        Button btnOpen = (Button) findViewById(R.id.btnSet);
        btnOpen.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Toast.makeText(getBaseContext(),
                        "Date selected:" + datePicker.getMonth() +
                        "/" + datePicker.getDayOfMonth() +
                        "/" + datePicker.getYear() + "\n" +
                        "Time selected:" + timePicker.getCurrentHour() +
                        ":" + timePicker.getCurrentMinute(),
                        Toast.LENGTH_SHORT).show();
                }
        });
    }

    @Override
    protected Dialog onCreateDialog(int id)
    {
        switch (id) {
            case TIME_DIALOG_ID:
                return new TimePickerDialog(
                    this, mTimeSetListener, hour, minute, false);
            case DATE_DIALOG_ID:
                return new DatePickerDialog(
                    this, mDateSetListener, yr, month, day);
        }
        return null;
    }

    private DatePickerDialog.OnDateSetListener mDateSetListener =
        new DatePickerDialog.OnDateSetListener()
        {
            public void onDateSet(
            DatePicker view, int year, int monthOfYear, int dayOfMonth)
            {
                yr = year;
                month = monthOfYear;
                day = dayOfMonth;
                Toast.makeText(getBaseContext(),
                        "You have selected : " + (month + 1) +
```

```
                    "/" + day + "/" + year,
                    Toast.LENGTH_SHORT).show();
            }
        };

    private TimePickerDialog.OnTimeSetListener mTimeSetListener =
        new TimePickerDialog.OnTimeSetListener()
        {
            public void onTimeSet(
            TimePicker view, int hourOfDay, int minuteOfHour)
            {
                hour = hourOfDay;
                minute = minuteOfHour;
                Toast.makeText(getBaseContext(),
                        "You have selected : " + hour + ":" + minute,
                        Toast.LENGTH_SHORT).show();

            }
        };
}
```

(2) 按F11键在Android模拟器上调试应用程序。当活动加载时，可以看到DatePicker显示在一个对话框窗口中(如图4-14所示)。设定好一个日期并单击Set按钮。Toast窗口将显示出您刚刚设置好的日期。

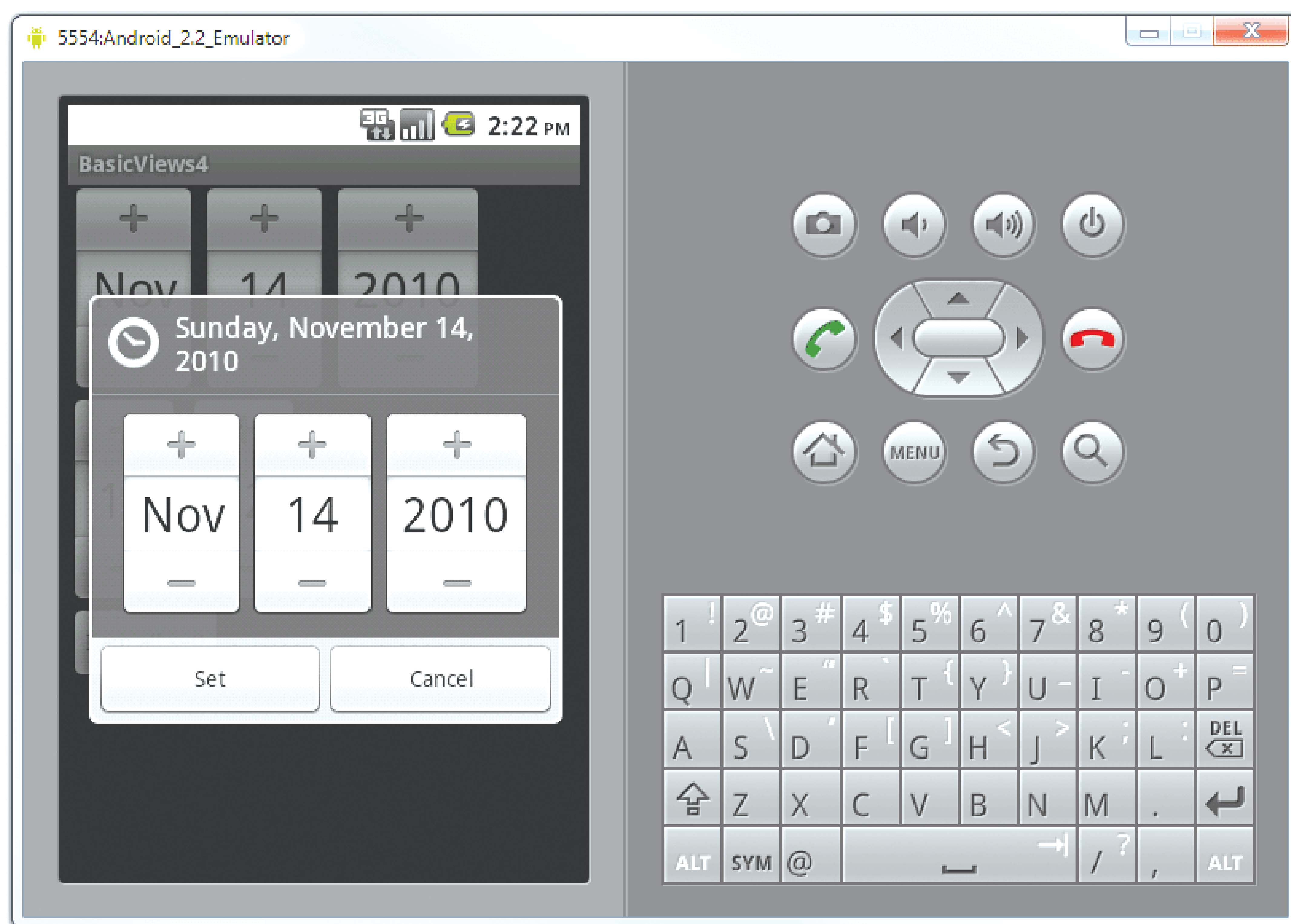


图 4-14

示例说明

DataPicker和TimePicker的工作原理是一致的。当设置日期时，它将触发onDateSet()方法，从中可以获取由用户设定的日期：

```
public void onDateSet(
DatePicker view, int year, int monthOfYear, int dayOfMonth)
{
    yr = year;
    month = monthOfYear;
    day = dayOfMonth;
    Toast.makeText(getBaseContext(),
            "You have selected : " + (month + 1) +
            "/" + day + "/" + year,
            Toast.LENGTH_SHORT).show();
}
```

注意，在显示对话框之前，需要初始化3个变量——yr、month和day：

```
//---获取当前日期---
Calendar today = Calendar.getInstance();
yr = today.get(Calendar.YEAR);
month = today.get(Calendar.MONTH);
day = today.get(Calendar.DAY_OF_MONTH);
showDialog(DATE_DIALOG_ID);
```

如果不这样做，当在运行时创建一个DatePickerDialog类的实例时，将发生非法参数异常(current should be >= start and <= end)。

4.3 列表视图

列表视图是一种可以用来显示长的项列表的视图。在Android中，有两种列表视图：ListView和SpinnerView，两者都用于显示长的项列表。下面的“试一试”展示了这两种视图的使用。

4.3.1 ListView视图

ListView在一个垂直滚动列表中显示项列表。下面的“试一试”演示了如何使用ListView显示一个项列表。

试一试 使用ListView显示一个长的项列表

BasicViews5.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个名为BasicViews5的Android项目。

(2) 修改MainActivity.java文件，插入下列粗体显示的语句：

```
package net.learn2develop.BasicViews5;

import android.app.Activity;
```

```
import android.os.Bundle;

import android.app.ListActivity;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends ListActivity {

    String[] presidents = {
        "Dwight D. Eisenhower",
        "John F. Kennedy",
        "Lyndon B. Johnson",
        "Richard Nixon",
        "Gerald Ford",
        "Jimmy Carter",
        "Ronald Reagan",
        "George H. W. Bush",
        "Bill Clinton",
        "George W. Bush",
        "Barack Obama"
    };

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.main);

        setListAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, presidents));
    }

    public void onListItemClick(
    ListView parent, View v, int position, long id)
    {
        Toast.makeText(this,
            "You have selected " + presidents[position],
            Toast.LENGTH_SHORT).show();
    }
}
```

(3) 按F11键在Android模拟器上调试应用程序。图4-15展示了显示总统名字列表的活动。

(4) 单击一个列表项，将显示一个包含所选择项的消息。

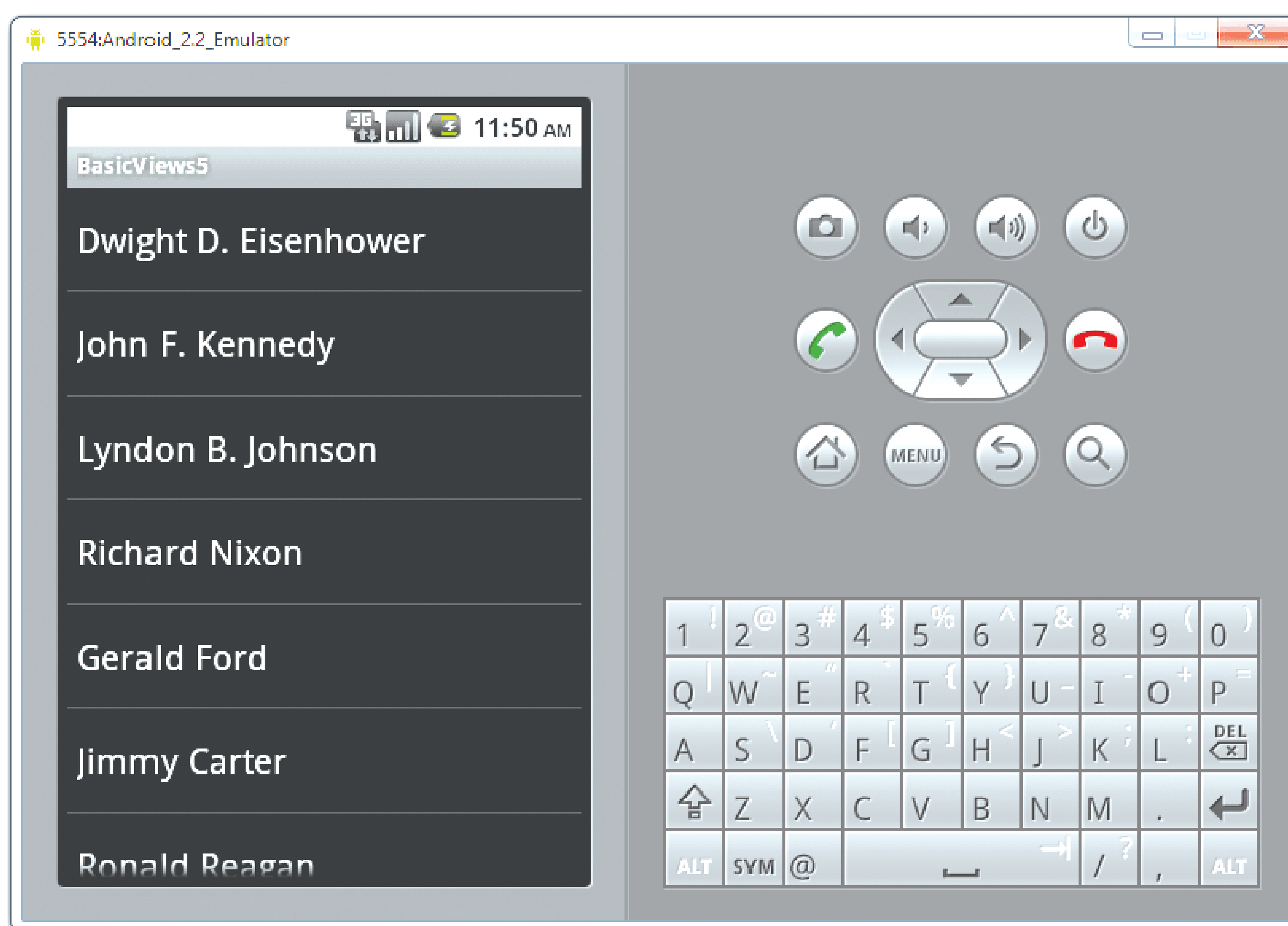


图 4-15

示例说明

在本例中，首先要注意的是MainActivity类扩展了ListActivity类。ListActivity类扩展了Activity类并且通过绑定到一个数据源来显示一个项列表。还要注意，无须修改main.xml文件来包含ListView；ListActivity类本身已经包含了一个ListView。因此，在onCreate()方法中，不需要调用setContentView()方法来从main.xml文件中加载用户界面：

```
//---不需要调用这个方法---
//setContentView(R.layout.main);
```

在onCreate()方法中，使用setListAdapter()方法来用一个ListView以编程方式填充活动的整个屏幕。ArrayAdapter对象管理将由ListView显示的字符串数组。在前面的例子中，将ListView设置为在simple_list_item_1模式下显示：

```
setListAdapter(new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, presidents));
```

当单击ListView中的一个列表项时，将触发onListItemClick()方法：

```
public void onListItemClick(
ListView parent, View v, int position, long id)
{
    Toast.makeText(this,
        "You have selected " + presidents[position],
        Toast.LENGTH_SHORT).show();
}
```

这里，只是使用Toast类来显示所选择的总统名字。

定制ListView

ListView是一个可以进一步定制的通用控件。下面的“试一试”展示了如何允许在ListView中选择多个项以及如何使之支持筛选功能。

试一试　定制ListView

(1) 打开前一节中创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
/** 当活动第一次被创建时调用。 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    ListView lstView = getListView();
    //lstView.setChoiceMode(0); //CHOICE_MODE_NONE
    //lstView.setChoiceMode(1); //CHOICE_MODE_SINGLE
    lstView.setChoiceMode(2);   //CHOICE_MODE_MULTIPLE
    lstView.setTextFilterEnabled(true);

    setListAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_checked, presidents));
}

public void onListItemClick(
ListView parent, View v, int position, long id)
{
    //---打上在列表项旁边显示的勾号---
    parent.setItemChecked(position, parent.isItemChecked(position));
    Toast.makeText(this,
        "You have selected " + presidents[position],
        Toast.LENGTH_SHORT).show();
}
```

(2) 按F11键在Android模拟器上调试应用程序。现在，可以单击每个项以显示其旁边的勾号图标(如图4-16所示)。

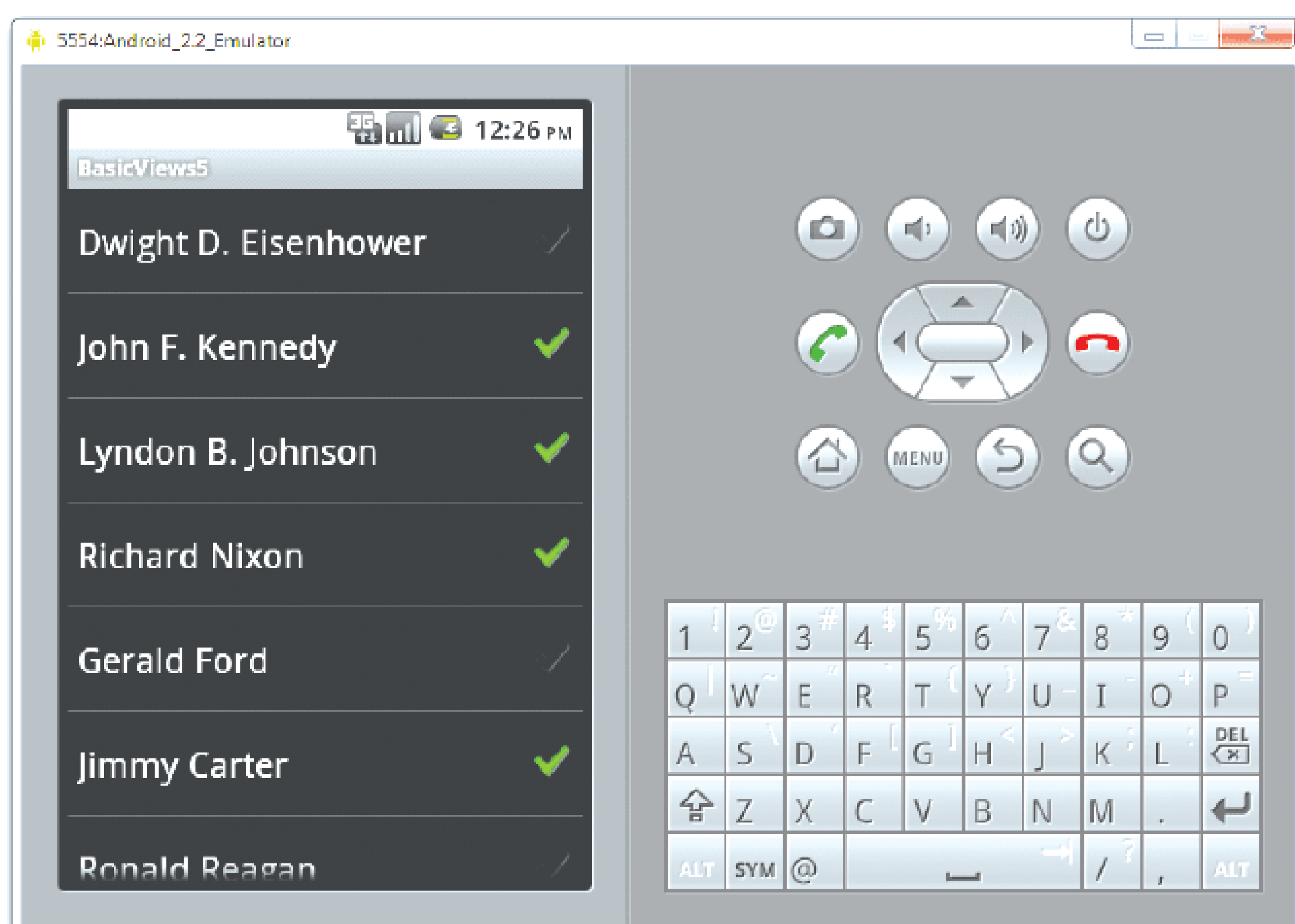


图 4-16

示例说明

为了以编程方式获得对ListView对象的引用，可以使用能获取ListActivity的列表视图的getListView()方法。想要以编程方式修改ListView的行为，就需要这么做。在此情况下，使用

setChoiceMode()方法来告诉ListView如何处理一个用户的单击。在本例中，设置其为2，这意味着用户可以选择多个项：

```
//lstView.setChoiceMode(0); //CHOICE_MODE_NONE
//lstView.setChoiceMode(1); //CHOICE_MODE_SINGLE
lstView.setChoiceMode(2);   //CHOICE_MODE_MULTIPLE
```

ListView的一个非常酷的功能是支持筛选。如果通过setTextFilterEnabled()方法启用了筛选功能，用户将可以在键盘上输入并且ListView将自动筛选来匹配已经输入的内容：

```
lstView.setTextFilterEnabled(true);
```

图4-17显示了起作用的列表筛选功能。这里，列表中所有包含单词john的项将在结果列表中显示出来。

为了显示每一个项旁边的勾号图标，使用setItemChecked()方法：

```
//---打上在列表项旁边显示的勾号---
parent.setItemChecked(position, parent.isItemChecked(position));
```

上述语句将在用户单击每一个项时，为其打上勾号图标。

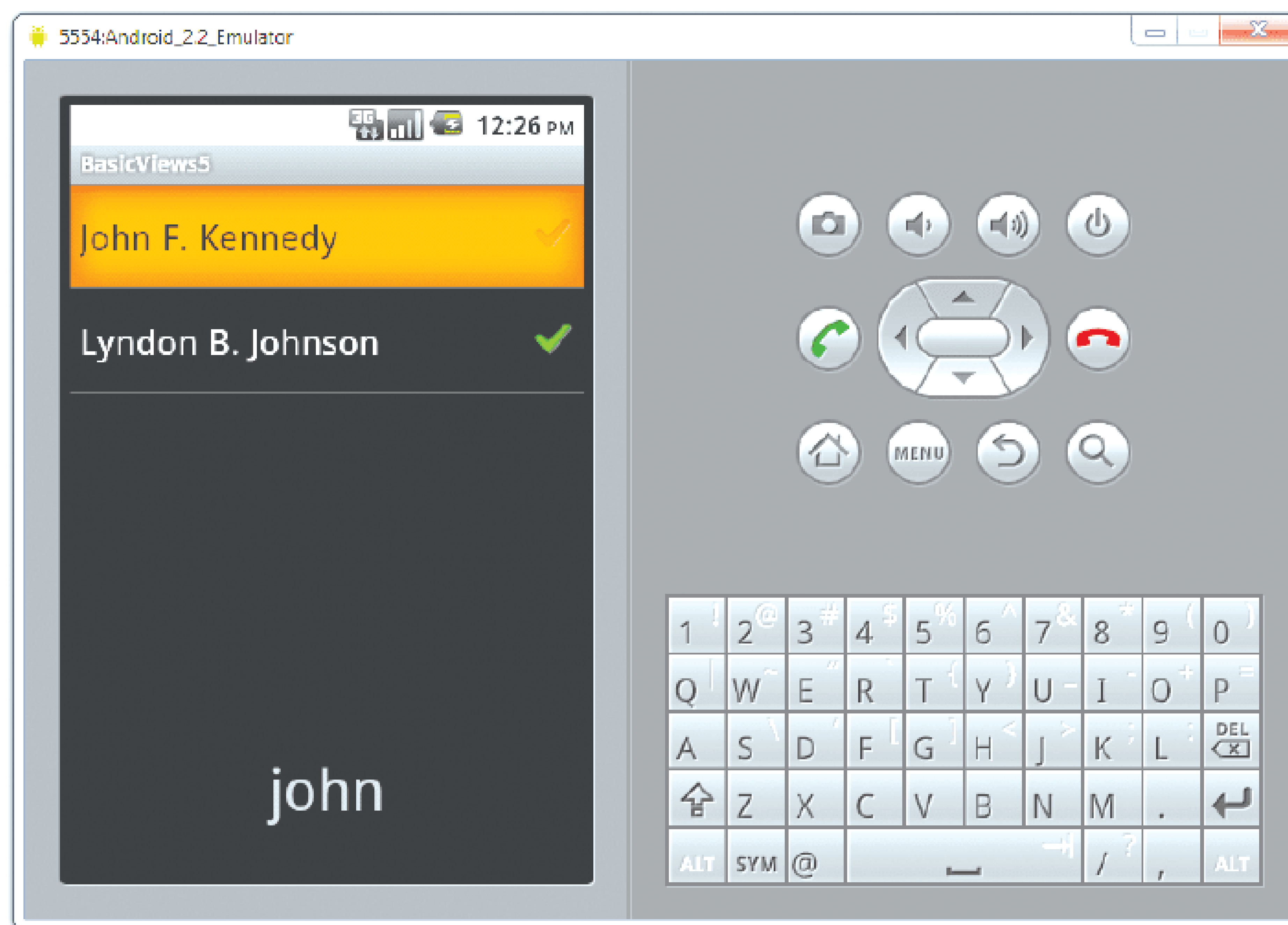


图 4-17

虽然本例中显示了总统名字列表存储在一个数组中，但在实际的应用中，建议从数据库中检索它们或至少将它们存储在strings.xml文件中。下面的“试一试”展示了这一点。

试一试　将列表项存储在strings.xml文件中

(1) 使用前一节创建的同一个项目，在位于res/values文件夹下的strings.xml文件中添加下列粗体显示的行：

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resources>
    <string name="hello">Hello World, MainActivity!</string>
    <string name="app_name">BasicViews5</string>
    <string-array name="presidents_array">
        <item>Dwight D. Eisenhower</item>
        <item>John F. Kennedy</item>
        <item>Lyndon B. Johnson</item>
        <item>Richard Nixon</item>
        <item>Gerald Ford</item>
        <item>Jimmy Carter</item>
        <item>Ronald Reagan</item>
        <item>George H. W. Bush</item>
        <item>Bill Clinton</item>
        <item>George W. Bush</item>
        <item>Barack Obama</item>
    </string-array>
</resources>
```

(2) 按下列粗体显示内容修改MainActivity.java文件：

```
public class MainActivity extends ListActivity {

    String[] presidents;

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        ListView lstView = getListView();
        //lstView.setChoiceMode(0); //CHOICE_MODE_NONE
        //lstView.setChoiceMode(1); //CHOICE_MODE_SINGLE
        lstView.setChoiceMode(2);   //CHOICE_MODE_MULTIPLE
        lstView.setTextFilterEnabled(true);

        presidents =
            getResources().getStringArray(R.array.presidents_array);

        setListAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_checked, presidents));
    }

    public void onListItemClick(
    ListView parent, View v, int position, long id)
    {
        //...
        //...
    }
};
```

(3) 按F11键在Android模拟器上调试应用程序。您将会看到同前面的“试一试”中一样的名字列表。

示例说明

由于现在名字存储在strings.xml文件中，所以可以在这个MainActivity.java文件中使用getResources()方法以编程方式来检索它：

```
presidents =
    getResources().getStringArray(R.array.presidents_array);
```

一般地，可以使用getResources()方法以编程方式来检索与应用程序捆绑的资源。

4.3.2 使用Spinner视图

ListView在一个活动中显示一个长的项列表，但有时需要在用户界面上显示其他视图，因此没有额外的空间来显示像ListView这样的全屏视图。在这种情况下，应该使用SpinnerView。SpinnerView一次显示列表中的一项，并可以使用户在其中进行选择。

下面的“试一试”展示了如何在活动中使用SpinnerView。

试一试 使用SpinnerView一次显示一个项

BasicViews6.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个名为BasicViews6的Android项目。

(2) 按如下所示修改位于res/layout文件夹下的main.xml文件：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

<Spinner
    android:id="@+id/spinner1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawSelectorOnTop="true" />

</LinearLayout>
```

(3) 把下列粗体显示的行添加到位于res/values文件夹下的strings.xml文件中：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, MainActivity!</string>
    <string name="app_name">BasicViews6</string>
```

```
    <string-array name="presidents_array">
        <item>Dwight D. Eisenhower</item>
        <item>John F. Kennedy</item>
        <item>Lyndon B. Johnson</item>
        <item>Richard Nixon</item>
        <item>Gerald Ford</item>
        <item>Jimmy Carter</item>
        <item>Ronald Reagan</item>
        <item>George H. W. Bush</item>
        <item>Bill Clinton</item>
        <item>George W. Bush</item>
        <item>Barack Obama</item>
    </string-array>
</resources>
```

(4) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.BasicViews6;

import android.app.Activity;
import android.os.Bundle;

import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

public class MainActivity extends Activity {

    String[] presidents;

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        presidents =
            getResources().getStringArray(R.array.presidents_array);
        Spinner s1 = (Spinner) findViewById(R.id.spinner1);

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_spinner_item, presidents);

        s1.setAdapter(adapter);
        s1.setOnItemSelectedListener(new OnItemSelectedListener()
        {
            @Override
            public void onItemSelected(AdapterView<?> arg0, View arg1,
```

```
                int arg2, long arg3)
            {
                int index = arg0.getSelectedItemPosition();
                Toast.makeText(getBaseContext(),
                    "You have selected item : " + presidents[index],
                    Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onNothingSelected(AdapterView<?> arg0) {}
        });
    }
}
```

(5) 按F11键在Android模拟器上调试应用程序。单击SpinnerView，可以看到弹出一个显示总统名字的列表(如图4-18所示)。单击一个列表项将显示一个消息，表明这个列表项被选择了。

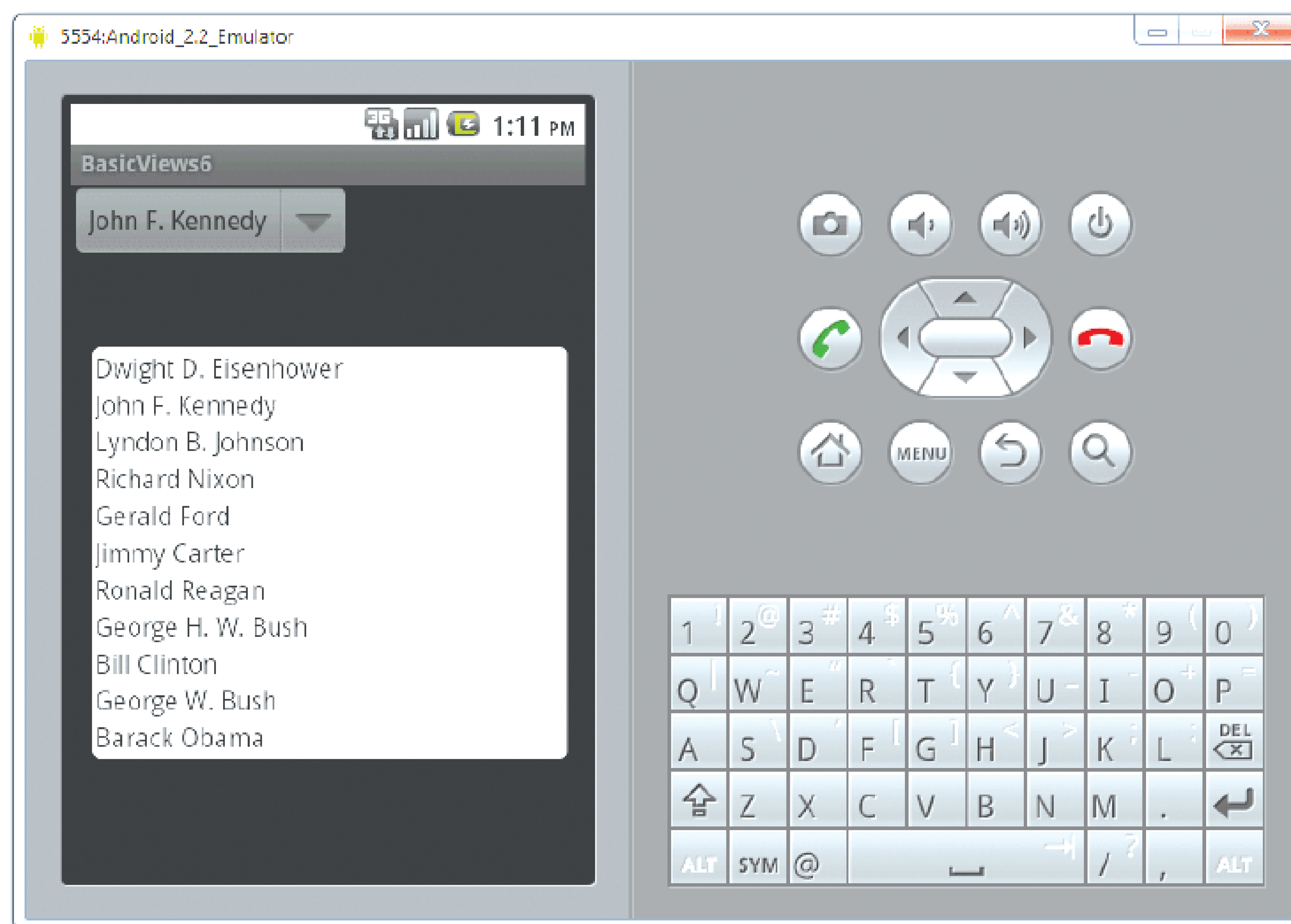


图 4-18

示例说明

上面的例子与ListView的工作原理很相像。需要实现的一个额外方法是onNothingSelected()方法。当用户按下Back按钮时触发这一方法，撤销所显示的项列表。在这种情况下，没有任何项被选择，也不需要作任何处理。

除了在ArrayAdapter中以普通列表形式显示列表项之外，还可以使用单选按钮来显示它们。要做到这一点，需要修改ArrayAdapter类的构造函数中的第二个参数：

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_spinner_dropdown_item, presidents);
```

这样将使列表项以单选按钮列表形式显示(如图4-19所示)。

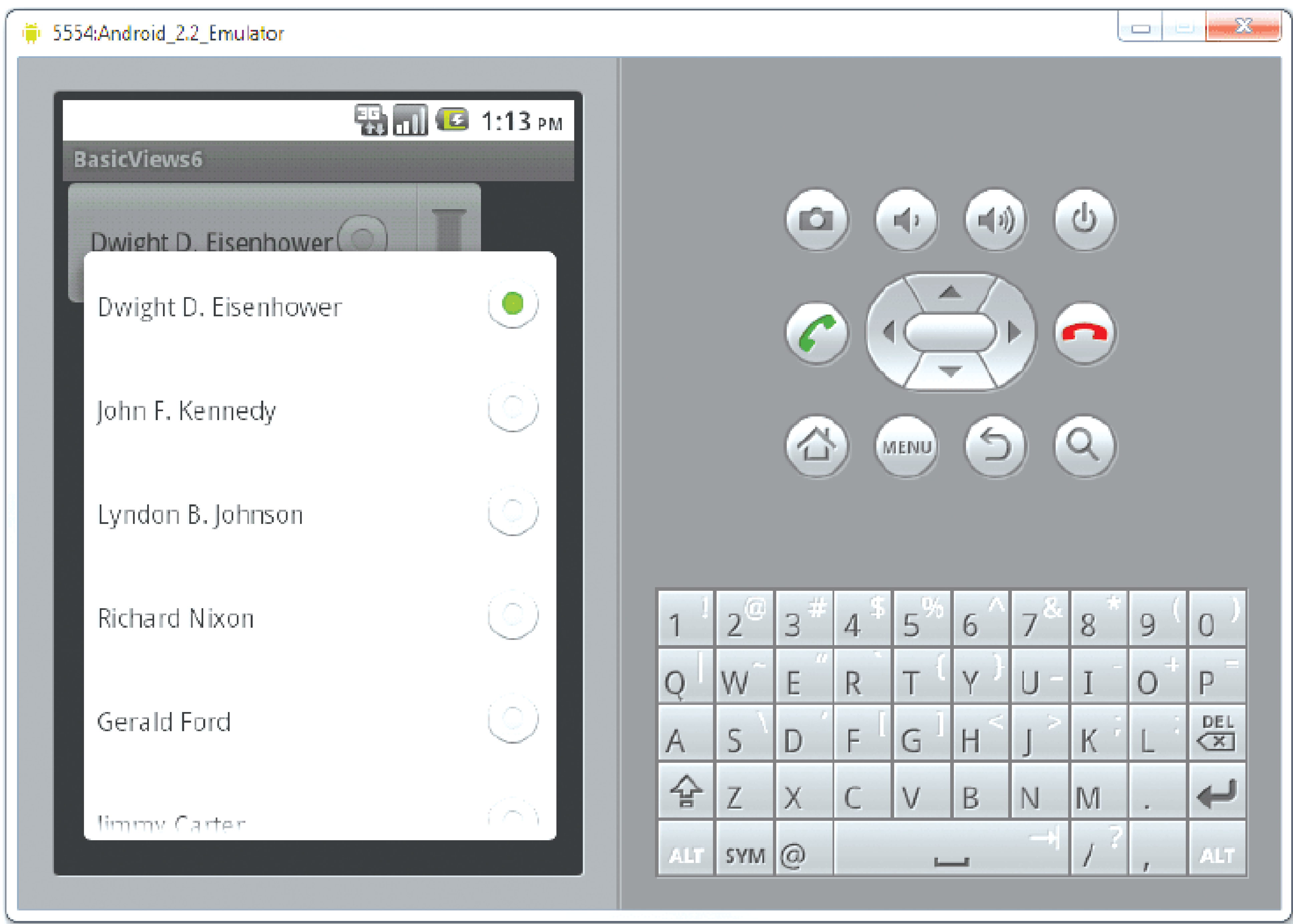


图 4-19

4.4　本章小结

本章对在Android应用程序中经常会用到的一些视图作了概述。虽然不可能详细研究每一个视图，但这里所学习到的视图会为设计Android应用程序的用户界面提供一个良好的基础，而不用管其需求是什么。

练　习

1. 如何以编程方式来确定一个RadioButton是否被选中?
2. 如何访问存储在strings.xml文件中的字符串资源?
3. 写一段代码来获取当前日期。

练习答案参见附录C。

本章主要内容

主　　题	关 键 概 念
TextView	`<TextView android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="@string/hello" />`

(续表)

主　　题	关 键 概 念
Button	<Button android:id="@+id/btnSave" android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="Save" />
ImageButton	<ImageButton android:id="@+id/btnImg1" android:layout_width="fill_parent" android:layout_height="wrap_content" android:src="@drawable/icon" />
EditText	<EditText android:id="@+id/txtName" android:layout_width="fill_parent" android:layout_height="wrap_content" />
CheckBox	<CheckBox android:id="@+id/chkAutosave" android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="Autosave" />
RadioGroup和RadioButton	<RadioGroup android:id="@+id/rdbGp1" android:layout_width="fill_parent" android:layout_height="wrap_content" android:orientation="vertical" > <RadioButton android:id="@+id/rdb1" android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="Option 1" /> <RadioButton android:id="@+id/rdb2" android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="Option 2" /> </RadioGroup>
ToggleButton	<ToggleButton android:id="@+id/toggle1" android:layout_width="wrap_content" android:layout_height="wrap_content" />
ProgressBar	<ProgressBar android:id="@+id/progressbar" android:layout_width="wrap_content" android:layout_height="wrap_content" />
AutoCompleteTextBox	<AutoCompleteTextView android:id="@+id/txtCountries" android:layout_width="fill_parent" android:layout_height="wrap_content" />
TimePicker	<TimePicker android:id="@+id/timePicker" android:layout_width="wrap_content" android:layout_height="wrap_content" />

(续表)

主　　题	关 键 概 念
DatePicker	<DatePicker android:id="@+id/datePicker" android:layout_width="wrap_content" android:layout_height="wrap_content" />
Spinner	<Spinner android:id="@+id/spinner1" android:layout_width="wrap_content" android:layout_height="wrap_content" android:drawSelectorOnTop="true" />

第5章 使用视图显示图片和菜单

本章将介绍以下内容

- 如何使用Gallery、ImageSwitcher、GridView和ImageView视图显示图像
- 如何显示选项菜单和上下文菜单
- 如何使用AnalogClock和DigitalClock视图显示时间
- 如何使用WebView显示Web内容

在第4章中，我们已经学习了可以用来构建Android应用程序的用户界面的不同视图。本章将继续研究其他可用来创建健壮的、吸引人的应用程序的视图。

特别是，我们会将注意力转到可以用来显示图像的视图上。此外，还将学习如何在Android应用程序中创建选项和上下文菜单。本章结束时将讨论一些可用来显示当前时间和Web内容的很炫酷的视图。

5.1 使用图像视图显示图片

到目前为止，我们已经学习的所有视图都是用来显示文本信息的。要显示图像，可以使用ImageView、Gallery、ImageSwitcher和 GridView 视图。下面将详细介绍每一个视图。

5.1.1 Gallery和ImageView视图

Gallery是一种用固定在中间位置的水平滚动列表显示列表项(如图像)的视图。图5-1展示了Gallery视图在显示一些图像时的外观效果：



图 5-1

下面的“试一试”展示了如何使用Gallery视图显示一组图像。

试一试　使用Gallery视图

Gallery.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，如图5-2所示创建一个新的Android项目。

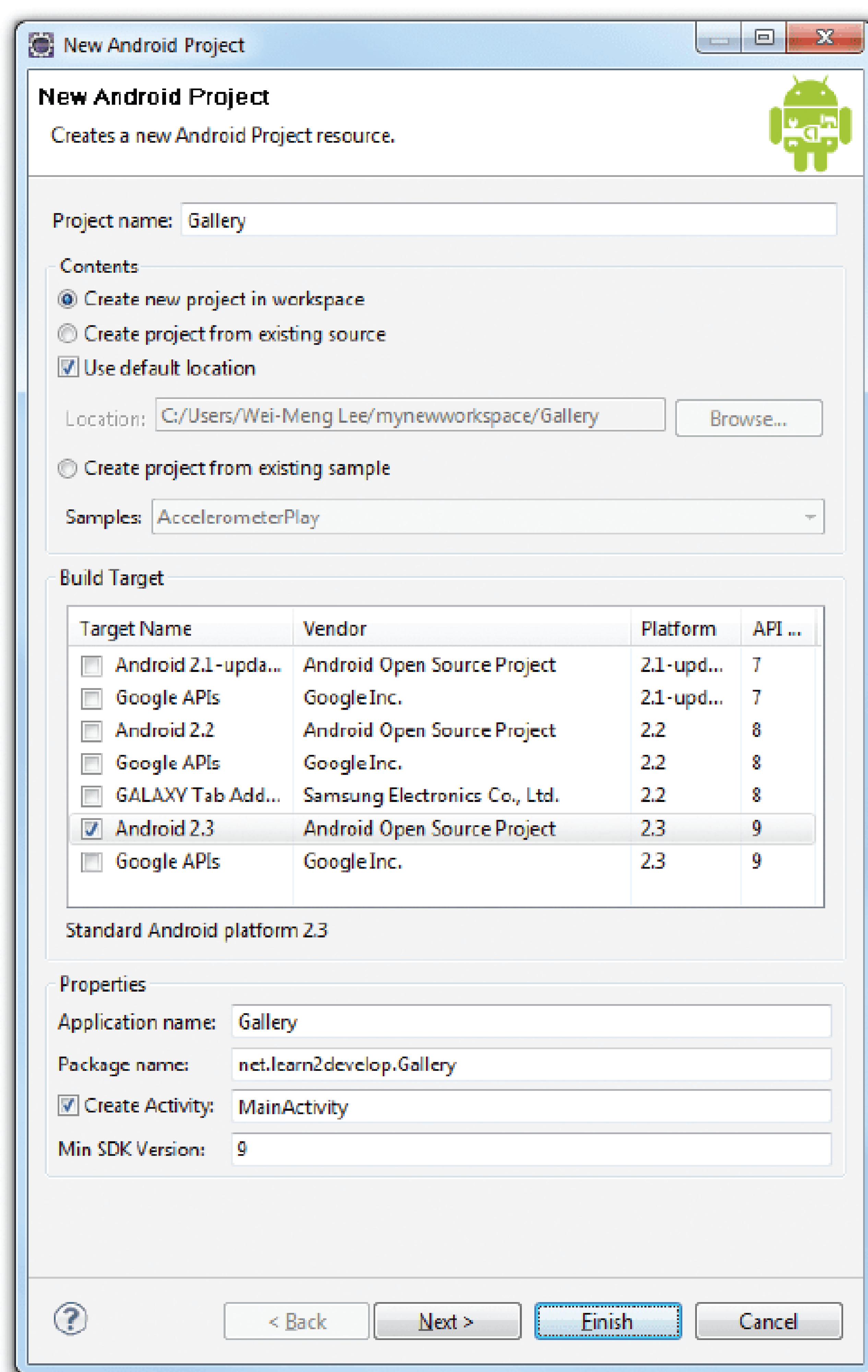


图 5-2

(2) 按照下列粗体显示内容修改main.xml文件：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Images of San Francisco" />

<Gallery
    android:id="@+id/gallery1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

```
<ImageView
    android:id="@+id/image1"
    android:layout_width="320px"
    android:layout_height="250px"
    android:scaleType="fitXY" />

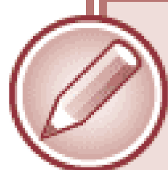
</LinearLayout>
```

(3) 右击res/values文件夹，选择New | File，并将新文件命名为attrs.xml。

(4) 在attrs.xml文件中输入如下内容：

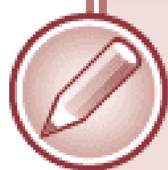
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="Gallery1">
        <attr name="android:galleryItemBackground" />
    </declare-styleable>
</resources>
```

(5) 准备一组图像，并将每一张图像依次命名为pic1.png、pic2.png等。



注意：可以从本书的支持网站www.wrox.com上下载这组图像。

(6) 将所有图像拖放到res/drawable-mdpi文件夹下(如图5-4所示)。当显示一个对话框时，选中复制选项并单击OK按钮。



注意：本例中假设这一项目将在具有中等DPI屏幕分辨率的AVD上进行测试。在实际的项目中，需要确保每一个drawable文件夹都有一组(不同分辨率的)图像。



图 5-3

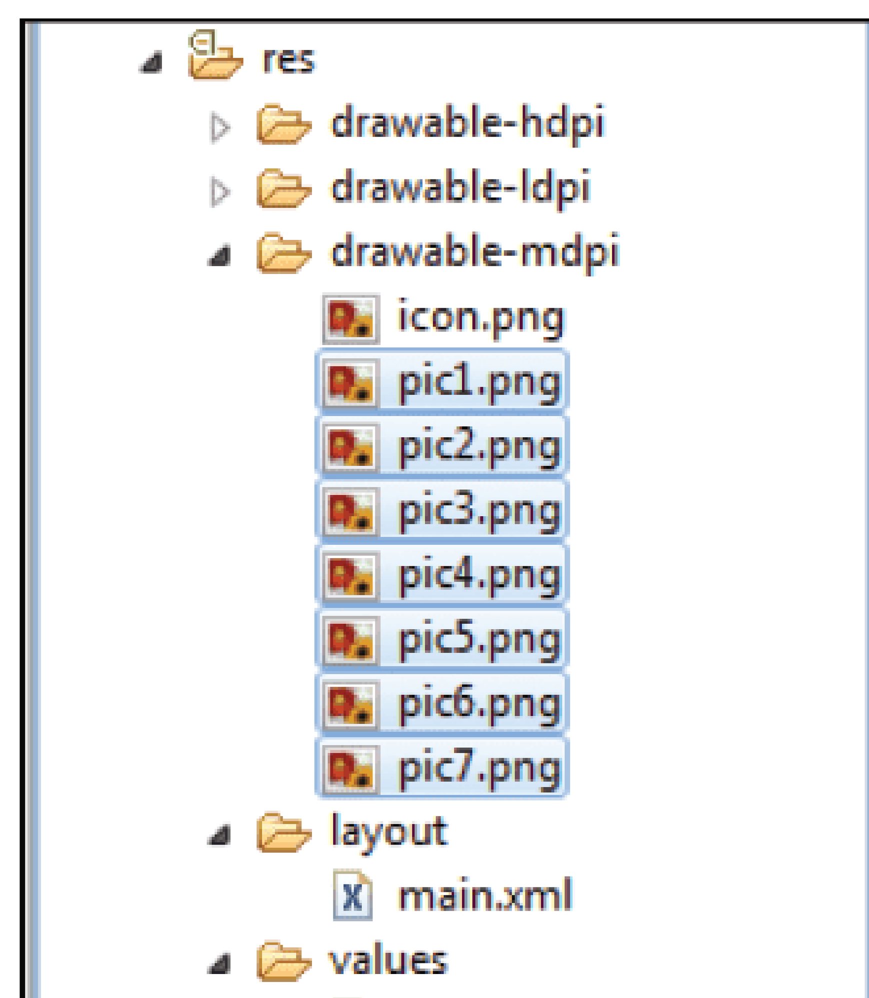


图 5-4

(7) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Gallery;

import android.app.Activity;
import android.os.Bundle;

import android.content.Context;
import android.content.res.TypedArray;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.Toast;

public class MainActivity extends Activity {
    //---要显示的图像---
    Integer[] imageIDs = {
            R.drawable.pic1,
            R.drawable.pic2,
            R.drawable.pic3,
            R.drawable.pic4,
            R.drawable.pic5,
            R.drawable.pic6,
            R.drawable.pic7
    };

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Gallery gallery = (Gallery) findViewById(R.id.gallery1);

        gallery.setAdapter(new ImageAdapter(this));
        gallery.setOnItemClickListener(new OnItemClickListener()
        {
            public void onItemClick(AdapterView<?> parent, View v,
            int position, long id)
            {
                Toast.makeText(getBaseContext(),
                        "pic" + (position + 1) + " selected",
                        Toast.LENGTH_SHORT).show();
            }
        });
    }
```

```
public class ImageAdapter extends BaseAdapter
{
    private Context context;
    private int itemBackground;

    public ImageAdapter(Context c)
    {
        context = c;
        //---设定样式---
        TypedArray a = obtainStyledAttributes(R.styleable.Gallery1);
        itemBackground = a.getResourceId(
            R.styleable.Gallery1_android_galleryItemBackground, 0);
        a.recycle();
    }

    //---返回图像数---
    public int getCount() {
        return imageIDs.length;
    }

    //---返回一个项的ID---
    public Object getItem(int position) {
        return position;
    }

     //---返回一个项的ID---
    public long getItemId(int position) {
        return position;
    }

    //---返回一个ImageView视图---
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView = new ImageView(context);
        imageView.setImageResource(imageIDs[position]);
        imageView.setScaleType(ImageView.ScaleType.FIT_XY);
        imageView.setLayoutParams(new Gallery.LayoutParams(150, 120));
        imageView.setBackgroundResource(itemBackground);
        return imageView;
    }
}
}
```

(8) 按F11键在Android模拟器上调试应用程序。图5-5展示了显示一系列图像的Gallery视图。

(9) 可以轻扫图像来显示整个系列的图像。单击一个图像时可以观察到Toast类将显示图像名称(如图5-6所示)。



图 5-5

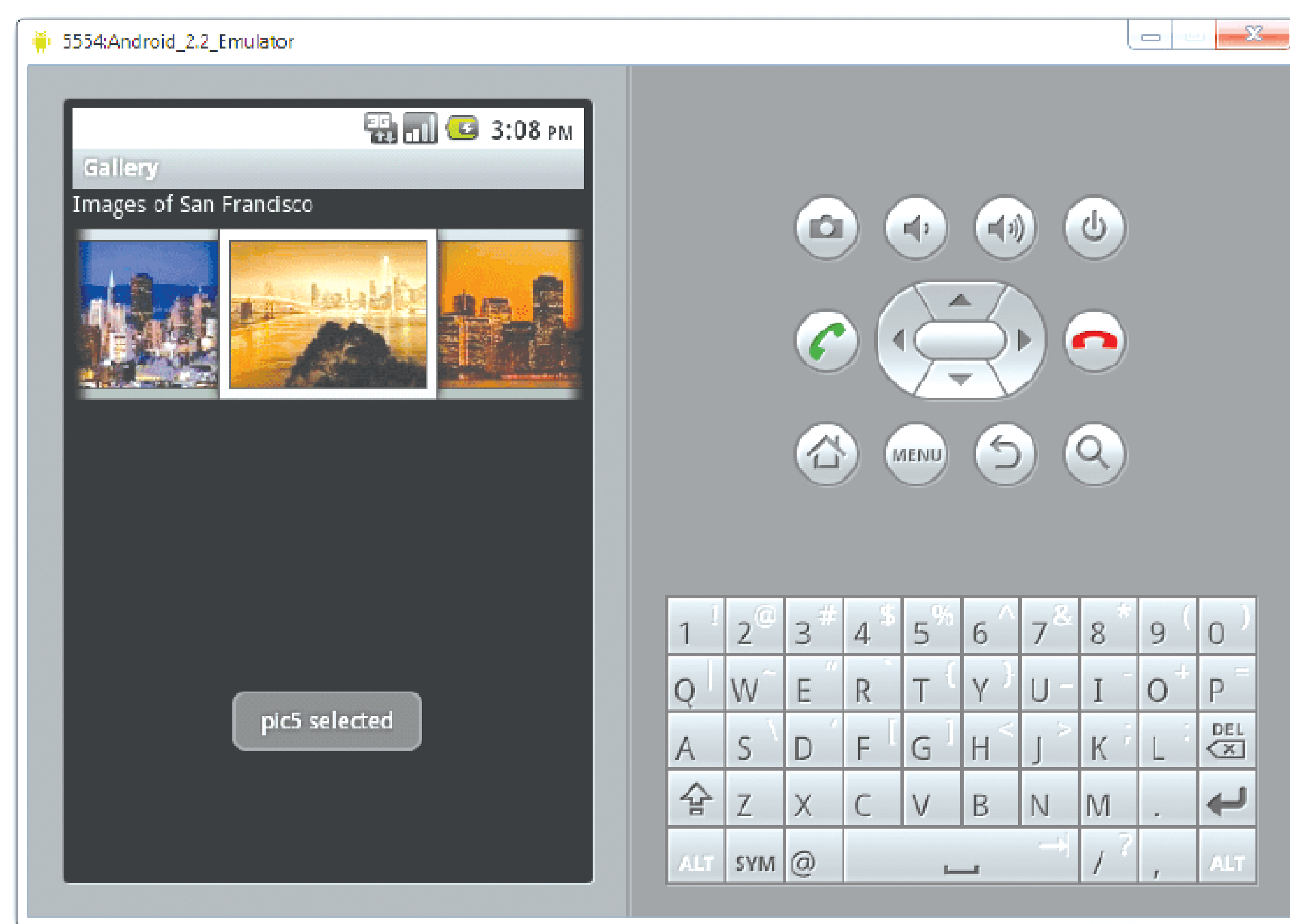


图 5-6

(10) 为了在ImageView中显示所选择的图像，在MainActivity.java文件中添加下列粗体显示的语句：

```
/** 当活动第一次被创建时调用。 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    Gallery gallery = (Gallery) findViewById(R.id.gallery1);

    gallery.setAdapter(new ImageAdapter(this));
```

```
            gallery.setOnItemClickListener(new OnItemClickListener()
            {
                public void onItemClick(AdapterView<?> parent, View v,
                int position, long id)
                {
                    Toast.makeText(getBaseContext(),
                            "pic" + (position + 1) + " selected",
                            Toast.LENGTH_SHORT).show();

                    //---显示选择的图像---
                    ImageView imageView = (ImageView) findViewById(R.id.image1);
                    imageView.setImageResource(imageIDs[position]);
                }
            });
        }
```

(11) 按F11键再次调试应用程序。这一次，将会看到在ImageView中显示了所选择的图像(如图5-7所示)。



图 5-7

示例说明

首先将Gallery和ImageView视图添加到main.xml文件中：

```
    <Gallery
        android:id="@+id/gallery1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <ImageView
        android:id="@+id/image1"
        android:layout_width="320px"
        android:layout_height="250px"
        android:scaleType="fitXY" />
```

正如先前所述，Gallery视图用来在一个水平滚动列表中显示一系列图像。ImageView用来显示用户选定的图像。

需要显示的图像列表存储在imageIDs数组中：

```
//---要显示的图像---
Integer[] imageIDs = {
        R.drawable.pic1,
        R.drawable.pic2,
        R.drawable.pic3,
        R.drawable.pic4,
        R.drawable.pic5,
        R.drawable.pic6,
        R.drawable.pic7
};
```

创建的ImageAdapter类扩展了BaseAdapter类，可以用来将一系列ImageView视图绑定到Gallery视图上：

```
Gallery gallery = (Gallery) findViewById(R.id.gallery1);
gallery.setAdapter(new ImageAdapter(this));
gallery.setOnItemClickListener(new OnItemClickListener()
{
    public void onItemClick(AdapterView<?> parent, View v,
    int position, long id)
    {
        Toast.makeText(getBaseContext(),
                "pic" + (position + 1) + " selected",
                Toast.LENGTH_SHORT).show();

        //---显示选择的图像---
        ImageView imageView = (ImageView) findViewById(R.id.image1);
        imageView.setImageResource(imageIDs[position]);
    }
});
```

当选定(也即单击)Gallery视图中的一张图像时，将显示出所选定图像的位置(第一张图像是0，第二张图像是1，依次类推)并在ImageView中显示此图像。

5.1.2　ImageSwitcher

前面一节演示了如何将Gallery视图与一个ImageView视图一起使用来显示一系列缩略图像，以便当其中之一被选中时，选定的图像在ImageView中显示。然而，有时您并不想当用户在Gallery视图中选择一张图像时该图像显示得太突然——例如，您也许希望在图像之间进行过渡时应用一些动画效果。这时，Gallery视图就需要ImageSwitcher来配合使用。下面的“试一试”展示了使用方法。

试一试 使用ImageSwitcher视图

ImageSwitcher.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个名为ImageSwitcher的Android项目。

(2) 修改main.xml文件，添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff000000" >

<Gallery
    android:id="@+id/gallery1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

<ImageSwitcher
    android:id="@+id/switcher1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true" />

</RelativeLayout>
```

(3) 右击res/values文件夹，选择New | File，并将文件命名为attrs.xml。

(4) 在attrs.xml文件中输入如下内容：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="Gallery1">
        <attr name="android:galleryItemBackground" />
    </declare-styleable>
</resources>
```

(5) 将一组图像拖放到res/drawable-mdpi文件夹下。当显示一个对话框时，选中复制选项并单击OK按钮。

(6) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.ImageSwitcher;

import android.app.Activity;
import android.os.Bundle;

import android.content.Context;
import android.content.res.TypedArray;
```

```
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewGroup.LayoutParams;
import android.view.animation.AnimationUtils;
import android.widget.BaseAdapter;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.Gallery;
import android.widget.ViewSwitcher.ViewFactory;
import android.widget.ImageSwitcher;
import android.widget.ImageView;

public class MainActivity extends Activity implements ViewFactory {
    //---要显示的图像---
    Integer[] imageIDs = {
            R.drawable.pic1,
            R.drawable.pic2,
            R.drawable.pic3,
            R.drawable.pic4,
            R.drawable.pic5,
            R.drawable.pic6,
            R.drawable.pic7
    };

    private ImageSwitcher imageSwitcher;

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        imageSwitcher = (ImageSwitcher) findViewById(R.id.switcher1);
        imageSwitcher.setFactory(this);
        imageSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.fade_in));
        imageSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.fade_out));

        Gallery gallery = (Gallery) findViewById(R.id.gallery1);
        gallery.setAdapter(new ImageAdapter(this));
        gallery.setOnItemClickListener(new OnItemClickListener()
        {
            public void onItemClick(AdapterView<?> parent,
            View v, int position, long id)
            {
                imageSwitcher.setImageResource(imageIDs[position]);
            }
        });
    }
```

```
    public View makeView()
    {
        ImageView imageView = new ImageView(this);
        imageView.setBackgroundColor(0xFF000000);
        imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
        imageView.setLayoutParams(new
                ImageSwitcher.LayoutParams(
                        LayoutParams.FILL_PARENT,
                        LayoutParams.FILL_PARENT));
        return imageView;
    }

    public class ImageAdapter extends BaseAdapter
    {
        private Context context;
        private int itemBackground;

        public ImageAdapter(Context c)
        {
            context = c;

            //---设定样式---
            TypedArray a = obtainStyledAttributes(R.styleable.Gallery1);
            itemBackground = a.getResourceId(
                    R.styleable.Gallery1_android_galleryItemBackground, 0);
            a.recycle();
        }

        //---返回图像数---
        public int getCount()
        {
            return imageIDs.length;
        }

        //---返回一个项的ID---
        public Object getItem(int position)
        {
            return position;
        }

        public long getItemId(int position)
        {
            return position;
        }

        //---返回一个ImageView视图---
        public View getView(int position, View convertView, ViewGroup parent)
        {
            ImageView imageView = new ImageView(context);
```

```
                imageView.setImageResource(imageIDs[position]);
                imageView.setScaleType(ImageView.ScaleType.FIT_XY);
                imageView.setLayoutParams(new Gallery.LayoutParams(150, 120));
                imageView.setBackgroundResource(itemBackground);

                return imageView;
            }
        }
    }
```

(7) 按F11键在Android模拟器上调试应用程序。图5-8展示了Gallery和ImageSwitcher视图，有两个图像集合以及选定的图像。



图 5-8

示例说明

在这个示例中，首先需要注意的是MainActivity不但扩展了Activity，而且还实现了ViewFactory。为了使用ImageSwitcher视图，需要实现ViewFactory接口，它创建了可与ImageSwitcher视图一起使用的视图。因此，需要实现makeView()方法：

```
public View makeView()
{
    ImageView imageView = new ImageView(this);
    imageView.setBackgroundColor(0xFF000000);
    imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
    imageView.setLayoutParams(new
            ImageSwitcher.LayoutParams(
                    LayoutParams.FILL_PARENT,
                    LayoutParams.FILL_PARENT));
    return imageView;
}
```

这个方法创建一个新的View来添加到ImageSwitcher视图中。

就像前一节的Gallery示例，需要实现一个ImageAdapter类来将一系列ImageView视图绑定到Gallery视图上。

在onCreate()方法中，可以获得对ImageSwitcher视图的引用并设置动画，指定图像应该如何“飞入”和“飞出”视图。最后，当在Gallery视图中选定一张图像时，它将在ImageSwitcher视图中显示出来：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    imageSwitcher = (ImageSwitcher) findViewById(R.id.switcher1);
    imageSwitcher.setFactory(this);
    imageSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.fade_in));
    imageSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.fade_out));

    Gallery gallery = (Gallery) findViewById(R.id.gallery1);
    gallery.setAdapter(new ImageAdapter(this));
    gallery.setOnItemClickListener(new OnItemClickListener()
    {
        public void onItemClick(AdapterView<?> parent,
        View v, int position, long id)
        {
            imageSwitcher.setImageResource(imageIDs[position]);
        }
    });
}
```

本例中，当在Gallery视图中选定一个图像时，它将以“淡入”的方式显示出来。当选定下一个图像时，当前图像将淡出。如果想让图像从左边滑入，而在选择另一幅图像时再从右边滑出，可尝试下面的动画效果：

```
imageSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.slide_in_left));
    imageSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.slide_out_right));
```

和前面的“试一试”一样，仍旧通过扩展BaseAdapter类来创建ImageAdapter类，以便将一系列ImageView视图绑定到Gallery视图上。

5.1.3 GridView

GridView在一个二维的滚动网格中来显示项。可以将GridView与一个ImageView配合使用来显示一系列图像。下面的“试一试”展示了如何做到这一点。

试一试　使用GridView视图

Grid.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个名为Grid的Android项目。

(2) 将一组图像拖放到res/drawable-mdpi文件夹下(如图5-9所示)。当显示一个对话框时，选中复制选项并单击OK按钮。

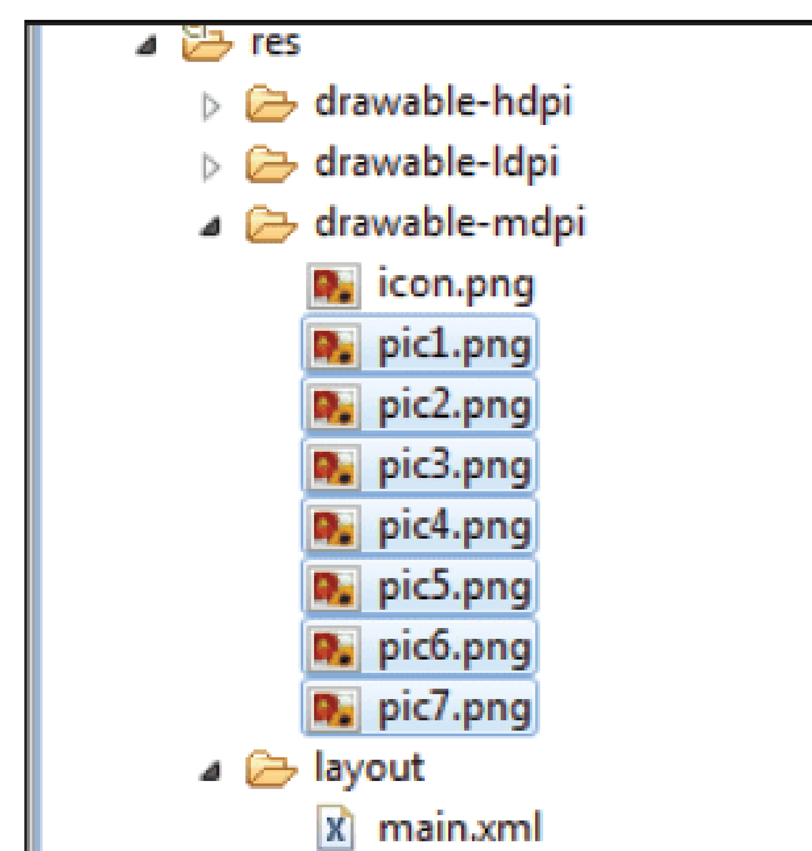


图 5-9

(3) 在main.xml文件中输入以下内容：

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.
com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:columnWidth="90dp"
    android:stretchMode="columnWidth"
    android:gravity="center"
/>
```

(4) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Grid;

import android.app.Activity;
import android.os.Bundle;

import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.Toast;

public class MainActivity extends Activity {
    //---要显示的图像---
    Integer[] imageIDs = {
            R.drawable.pic1,
            R.drawable.pic2,
            R.drawable.pic3,
            R.drawable.pic4,
            R.drawable.pic5,
            R.drawable.pic6,
```

```
        R.drawable.pic7
};

/** 当活动第一次被创建时调用。 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    GridView gridView = (GridView) findViewById(R.id.gridview);
    gridView.setAdapter(new ImageAdapter(this));

    gridView.setOnItemClickListener(new OnItemClickListener()
    {
        public void onItemClick(AdapterView<?> parent,
        View v, int position, long id)
        {
            Toast.makeText(getBaseContext(),
                    "pic" + (position + 1) + " selected",
                    Toast.LENGTH_SHORT).show();
        }
    });
}

public class ImageAdapter extends BaseAdapter
{
    private Context context;

    public ImageAdapter(Context c)
    {
        context = c;
    }

    //---返回图像数---
    public int getCount() {
        return imageIDs.length;
    }

    //---返回一个项的ID---
    public Object getItem(int position) {
        return position;
    }

    //---返回一个项的ID---
    public long getItemId(int position) {
        return position;
    }

    //---返回一个ImageView视图---
    public View getView(int position, View convertView,
```

```
                ViewGroup parent)
            {
                ImageView imageView;
                if (convertView == null) {
                    imageView = new ImageView(context);
                    imageView.setLayoutParams(new
                        GridView.LayoutParams(85, 85));
                    imageView.setScaleType(
                        ImageView.ScaleType.CENTER_CROP);
                    imageView.setPadding(5, 5, 5, 5);
                } else {
                    imageView = (ImageView) convertView;
                }
                imageView.setImageResource(imageIDs[position]);
                return imageView;
            }
        }
    }
```

(5) 按F11键在Android模拟器中调试应用程序。图5-10展示了显示所有图像的GridView。

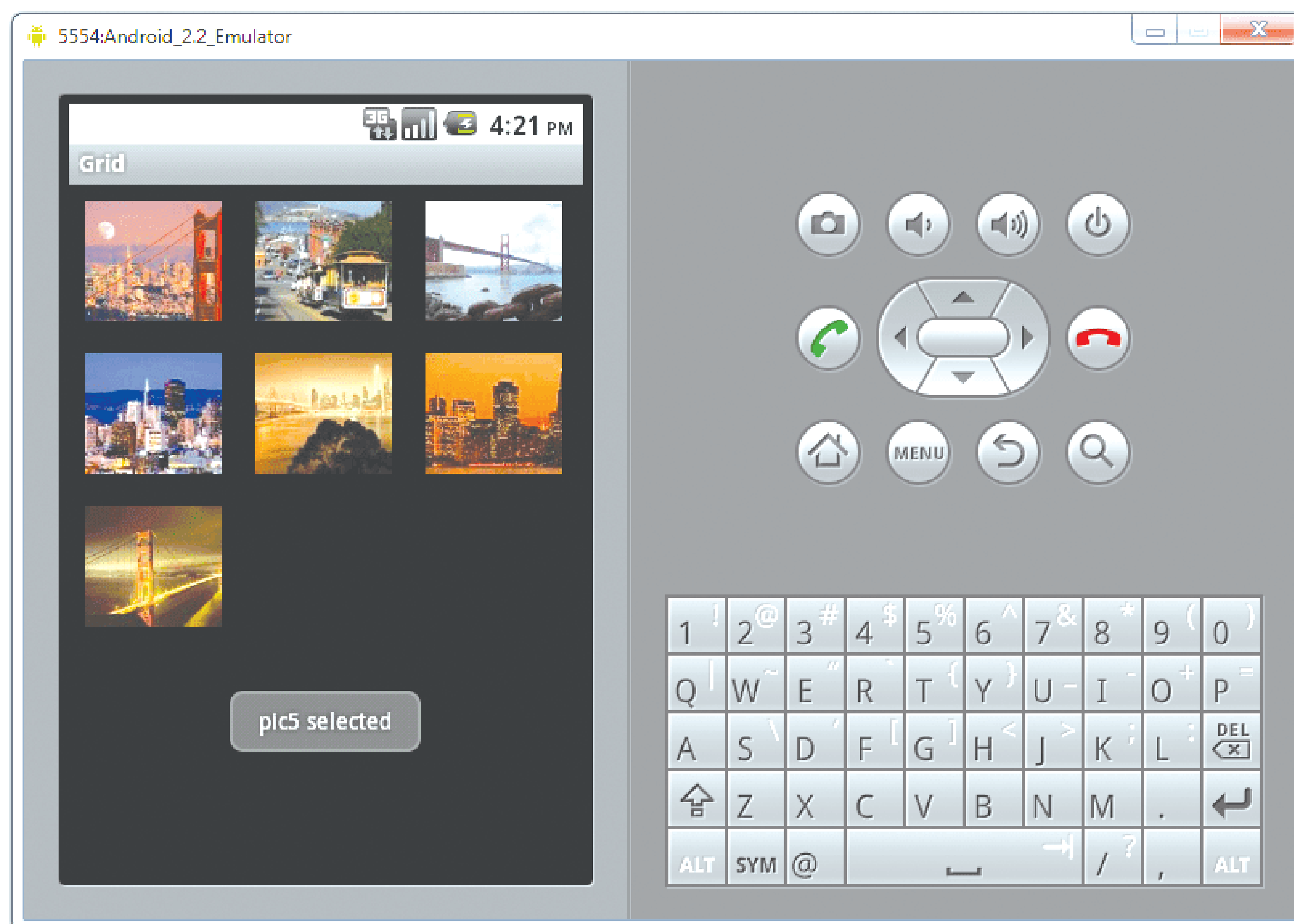


图 5-10

示例说明

与Gallery和ImageSwitcher示例一样，实现ImageAdapter类并绑定到GridView:

```
GridView gridView = (GridView) findViewById(R.id.gridview);
gridView.setAdapter(new ImageAdapter(this));

gridView.setOnItemClickListener(new OnItemClickListener()
{
```

```
            public void onItemClick(AdapterView<?> parent,
            View v, int position, long id)
            {
                Toast.makeText(getBaseContext(),
                        "pic" + (position + 1) + " selected",
                        Toast.LENGTH_SHORT).show();
            }
        });
```

当选择了一个图像时，将显示一个Toast消息表明该图像已被选定。

在GridView中，可以指定图像的大小，并通过为每幅图像设置内边距在GridView中对图像进行分隔：

```
        //---返回一个ImageView视图---
        public View getView(int position, View convertView,
        ViewGroup parent)
        {
            ImageView imageView;
            if (convertView == null) {
                imageView = new ImageView(context);
                imageView.setLayoutParams(new
                    GridView.LayoutParams(85, 85));
                imageView.setScaleType(
                    ImageView.ScaleType.CENTER_CROP);
                imageView.setPadding(5, 5, 5, 5);
            } else {
                imageView = (ImageView) convertView;
            }
            imageView.setImageResource(imageIDs[position]);
            return imageView;
        }
```

5.2 将菜单和视图一起使用

菜单用来显示在一个应用程序的主用户界面中不是直接可见的额外选项。在Android中有两种主要的菜单类型：

- 选项菜单——显示和当前活动相关的信息。在Android中，通过按下MENU键来激活选项菜单。
- 上下文菜单——显示和活动中一个特定的视图相关的信息。在Android中，通过单击并按住视图来激活上下文菜单。

图5-11展示了浏览器应用程序中的一个选项菜单的示例。当用户按下MENU按钮时，选项菜单将显示出来。所显示的菜单项随当前正在运行的活动而各异。

图5-12展示了当用户按下并保持在页面上的一个超链接时所显示的上下文菜单。显示的菜单项随当前选定的组件或视图而各异。为了激活上下文菜单，用户可在屏幕上选择一项，单击并按住它，或者直接按方向键盘上的中间按钮。



图 5-11

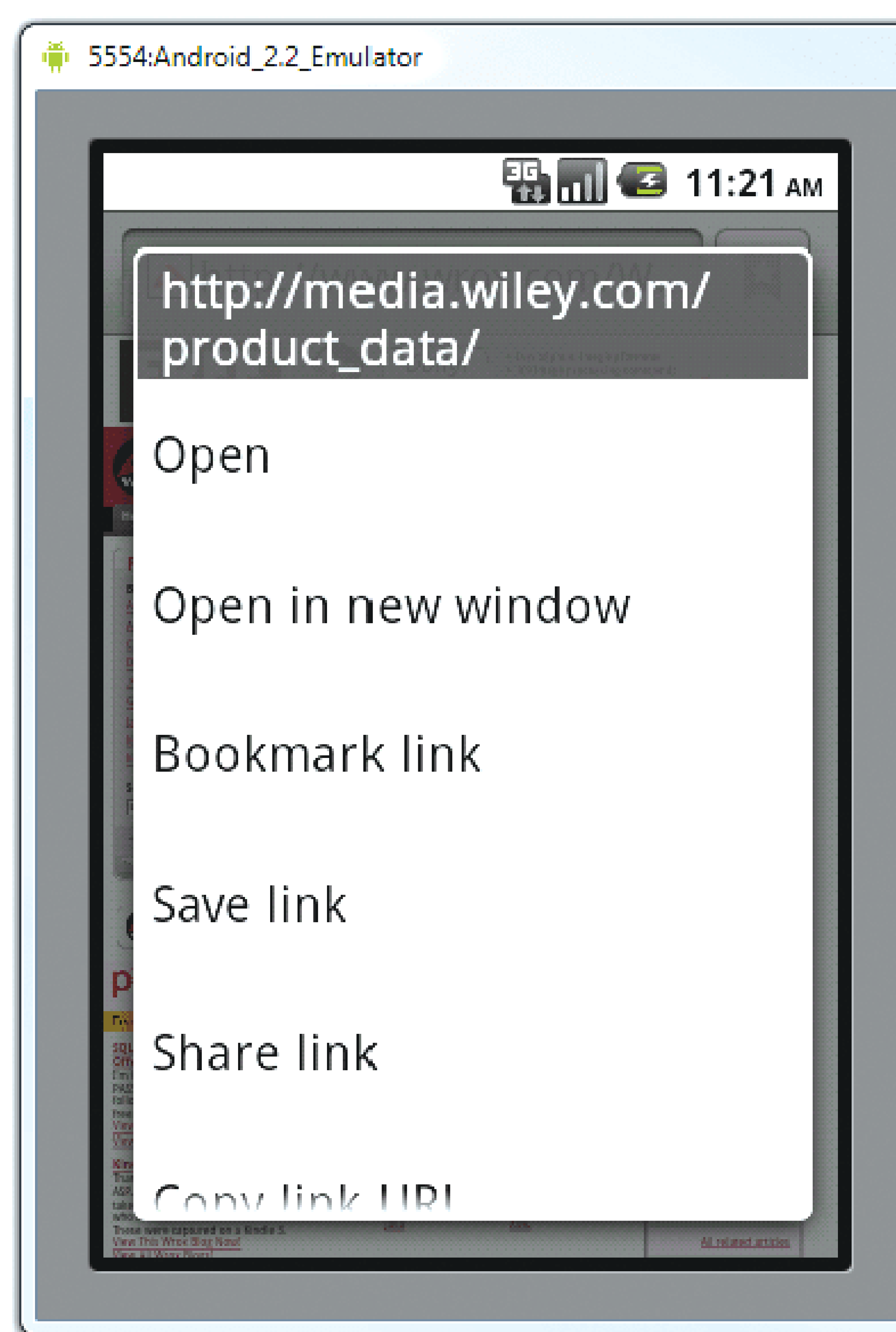


图 5-12

5.2.1　创建辅助方法

在深入学习和创建选项菜单以及上下文菜单之前，需要创建两个辅助方法。一个创建将在菜单内显示的项列表，另一个处理用户在菜单内选定一项时所触发的事件。

试一试　创建菜单辅助方法

Menus.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个名为Menus的Android项目。

(2) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Menus;

import android.app.Activity;
import android.os.Bundle;

import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.main);
    }

    private void CreateMenu(Menu menu)
    {
        MenuItem mnu1 = menu.add(0, 0, 0, "Item 1");
        {
            mnu1.setAlphabeticShortcut('a');
            mnu1.setIcon(R.drawable.icon);
        }
        MenuItem mnu2 = menu.add(0, 1, 1, "Item 2");
        {
            mnu2.setAlphabeticShortcut('b');
            mnu2.setIcon(R.drawable.icon);
        }
        MenuItem mnu3 = menu.add(0, 2, 2, "Item 3");
        {
            mnu3.setAlphabeticShortcut('c');
            mnu3.setIcon(R.drawable.icon);
        }
        MenuItem mnu4 = menu.add(0, 3, 3, "Item 4");
        {
            mnu4.setAlphabeticShortcut('d');
        }
        menu.add(0, 3, 3, "Item 5");
        menu.add(0, 3, 3, "Item 6");
        menu.add(0, 3, 3, "Item 7");
    }

    private boolean MenuChoice(MenuItem item)
    {
        switch (item.getItemId()) {
        case 0:
            Toast.makeText(this, "You clicked on Item 1",
                Toast.LENGTH_LONG).show();
            return true;
        case 1:
            Toast.makeText(this, "You clicked on Item 2",
                Toast.LENGTH_LONG).show();
            return true;
        case 2:
            Toast.makeText(this, "You clicked on Item 3",
                Toast.LENGTH_LONG).show();
            return true;
        case 3:
            Toast.makeText(this, "You clicked on Item 4",
                Toast.LENGTH_LONG).show();
            return true;
        case 4:
            Toast.makeText(this, "You clicked on Item 5",
```

```
                Toast.LENGTH_LONG).show();
            return true;
        case 5:
            Toast.makeText(this, "You clicked on Item 6",
                Toast.LENGTH_LONG).show();
            return true;
        case 6:
            Toast.makeText(this, "You clicked on Item 7",
                Toast.LENGTH_LONG).show();
            return true;
        }
        return false;
    }
}
```

示例说明

前面的示例创建了两个方法：CreateMenu()和MenuChoice()。CreatMenu()方法接受一个Menu参数并向其添加一系列菜单项。

为了向菜单中添加菜单项，需要创建一个MenuItem类的实例并使用Menu对象的add()方法。

```
MenuItem mnu1 = menu.add(0, 0, 0, "Item 1");
{
    mnu1.setAlphabeticShortcut('a');
    mnu1.setIcon(R.drawable.icon);
}
```

add()方法的4个参数如下所示：

- grouId——菜单项所在的组的标识符，使用0表示一个菜单项不在一个组中
- itemId——唯一的菜单项ID
- order——菜单项显示的顺序
- title——菜单项显示的文本

可以使用setAlphabeticShortcut()方法来给菜单项分配快捷键，这样用户可以通过在键盘上按键来选择一个菜单项。setIcon()方法设置将在菜单项上显示的图像。

MenuChoice()方法接受一个MenuItem参数，并检查其ID来确定被单击的菜单项。然后，它显示一个Toast消息告诉用户哪一个菜单项被单击了。

5.2.2　选项菜单

现在准备修改应用程序，使得当用户在Android设备上按下MENU按钮时显示选项菜单。

试一试　显示选项菜单

(1) 使用前一节所创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Menus;

import android.app.Activity;
```

```
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        CreateMenu(menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
         return MenuChoice(item);
    }

    private void CreateMenu(Menu menu)
    {
        //...
    }

    private boolean MenuChoice(MenuItem item)
    {
        //...
    }
}
```

(2) 按F11键在Android模拟器中调试应用程序。图5-13展示了当单击MENU按钮时所弹出的选项菜单。要选择一个菜单项，可以单击单独的菜单项或者使用其快捷键(A到D；只对前4个菜单项有效)。

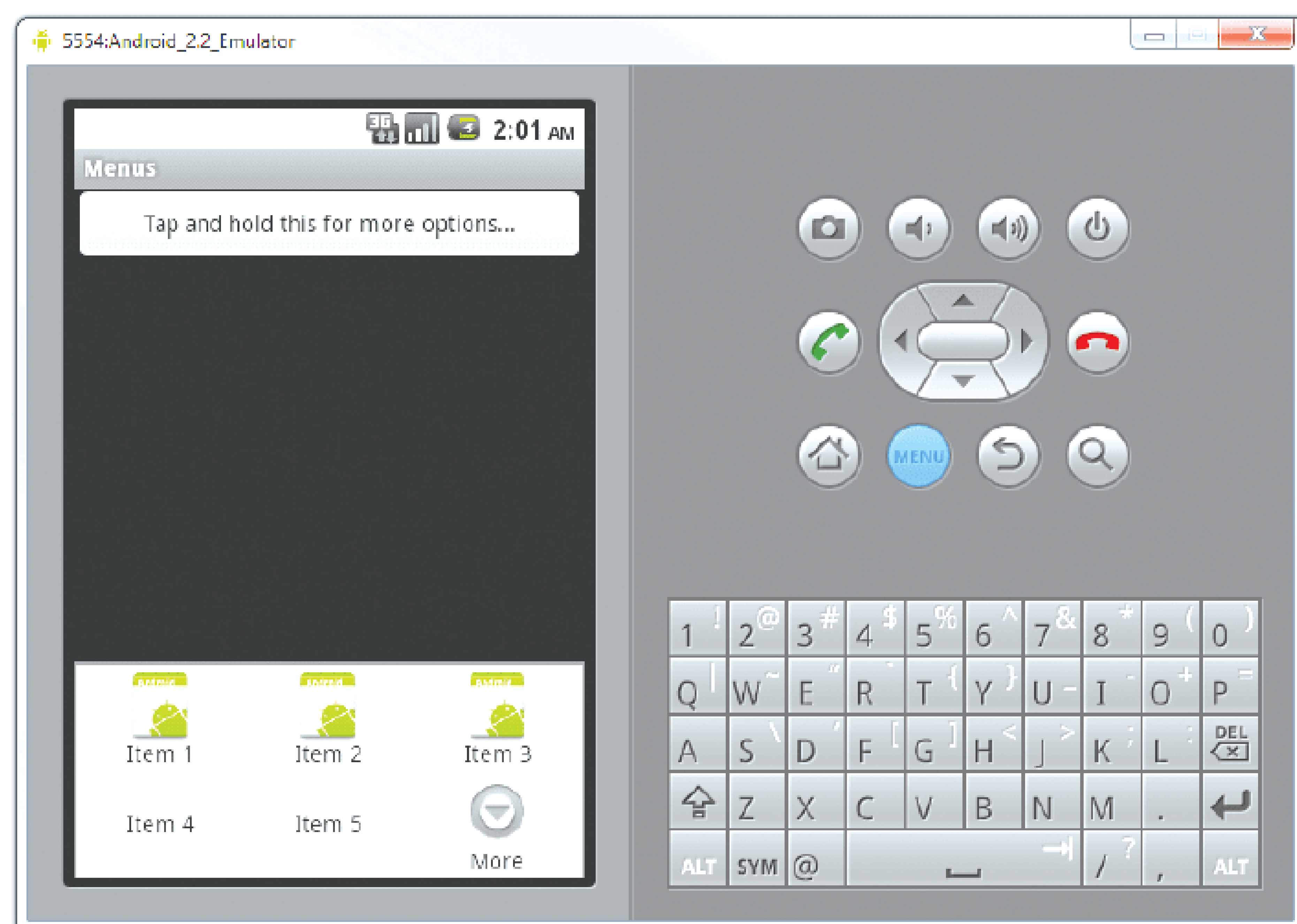


图 5-13

示例说明

为了显示活动的选项菜单，需要在活动中重写两个方法：onCreateOptionsMenu()和onOptionsItemSelected()。当MENU按钮被按下时调用onCreateOptionsMenu()方法。在这一事件中，调用CreateMenu()辅助方法来显示选项菜单。当选择了一个菜单项时，将调用onOptionsItemSelected()方法。这时，调用MenuChoice()方法来显示所选择的菜单项(并做任何想做的事)。

观察为菜单项1、2和3所显示的图标。如果选项菜单的菜单项数目超过6个，将会显示一个More菜单项来表明还有其他的选项。图5-14展示了在用户按下More菜单项后所显示的一个额外菜单项的列表。

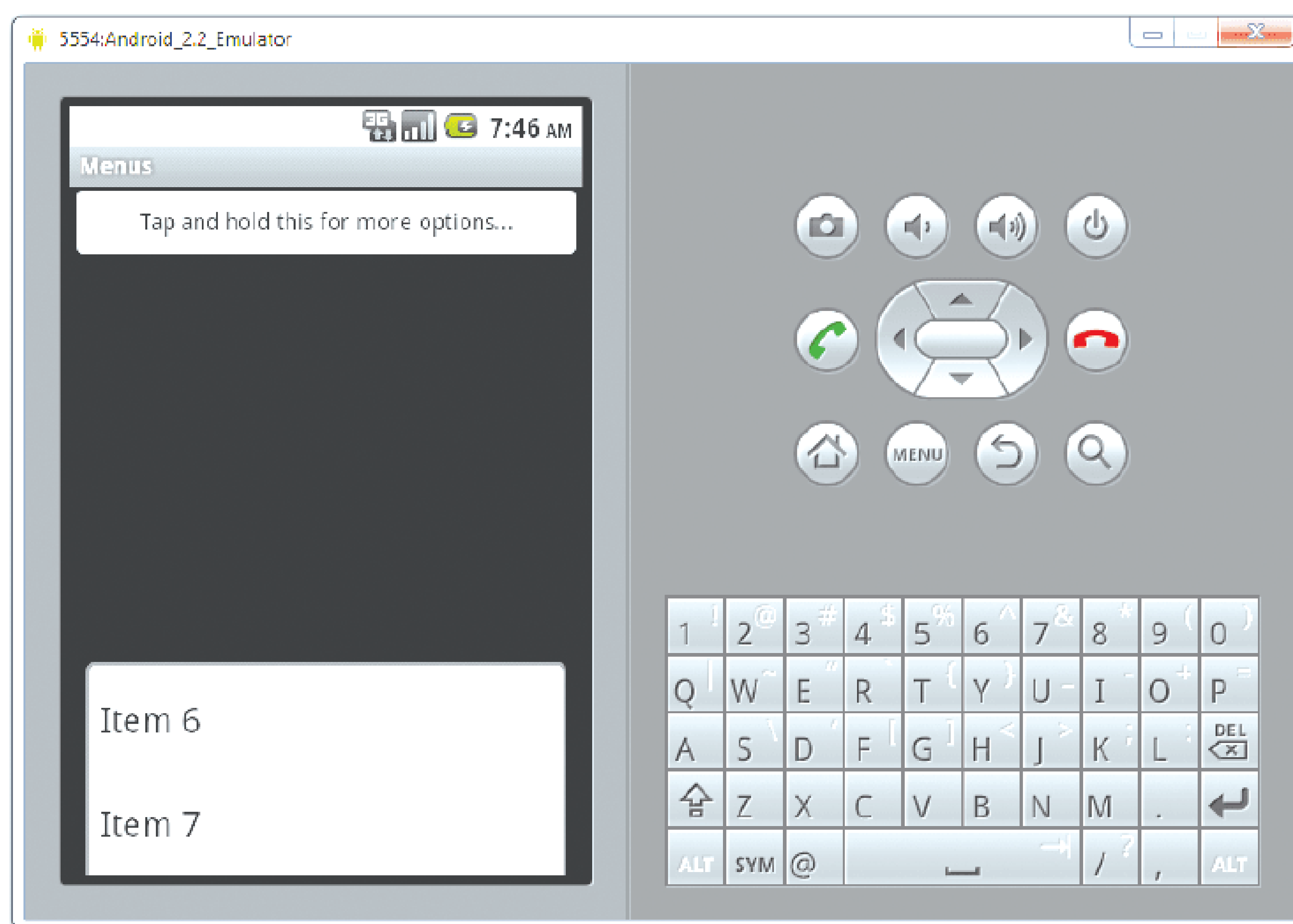


图 5-14

5.2.3　上下文菜单

前一节讲述了在用户按下MENU按钮时是如何显示选项菜单的。除了选项菜单，还可以显示上下文菜单。上下文菜单通常和活动上的一个视图相关联，并在用户长按一个项时显示。例如，如果用户按住Button视图并持续几秒钟，就会显示一个上下文菜单。

如果打算将上下文菜单和活动上的一个视图联系起来，需要调用那个视图的setOnCreateContextMenuListener()方法。下面的“试一试”展示了如何使一个上下文菜单与一个Button视图进行关联。

试一试　显示上下文菜单

(1) 使用前一节所创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Menus;

import android.app.Activity;
```

```
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.Toast;

import android.view.View;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button btn = (Button) findViewById(R.id.btn1);
        btn.setOnCreateContextMenuListener(this);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        CreateMenu(menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
         return MenuChoice(item);
    }

    @Override
    public void onCreateContextMenu(ContextMenu menu, View view,
    ContextMenuInfo menuInfo)
    {
         super.onCreateContextMenu(menu, view, menuInfo);
         CreateMenu(menu);
    }

    @Override
    public boolean onContextItemSelected(MenuItem item)
    {
         return MenuChoice(item);
    }

    private void CreateMenu(Menu menu)
    {
```

```
        //...
    }

    private boolean MenuChoice(MenuItem item)
    {
        //...
    }
}
```

(2) 按F11键在Android模拟器上调试应用程序。图5-15展示了在长按Button视图时所显示的上下文菜单。

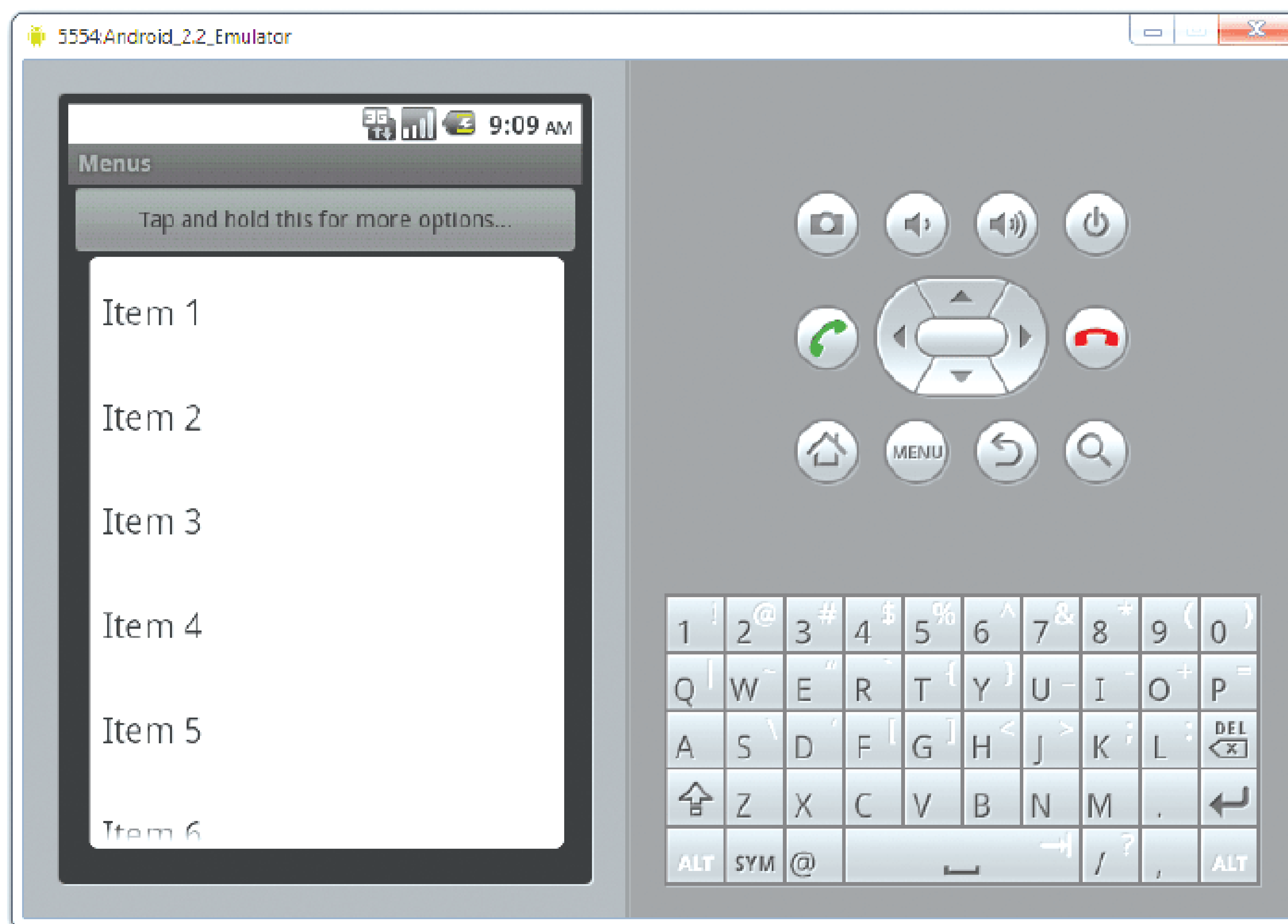


图 5-15

示例说明

在上述示例中，调用Button视图的setOnCreateContextMenuListener()方法将它和一个上下文菜单建立关联。

当用户长按Button视图时，onCreateContextMenu()方法将被调用。在这一方法中，通过调用CreateMenu()方法来显示上下文菜单。类似地，当上下文菜单内的一个项被选中时，onContextItemSelected()方法将被调用，在其中调用MenuChoice()方法来向用户显示一条消息。

注意，菜单项的快捷键并未生效。要使之生效，需要调用Menu对象的setQuertyMode()方法，如下所示：

```
    private void CreateMenu(Menu menu)
    {
        menu.setQwertyMode(true);
        MenuItem mnu1 = menu.add(0, 0, 0, "Item 1");
        {
            mnu1.setAlphabeticShortcut('d');
            mnu1.setIcon(R.drawable.icon);
        }
        //...
    }
```

这样做了以后就会使快捷键生效(如图5-16所示)。

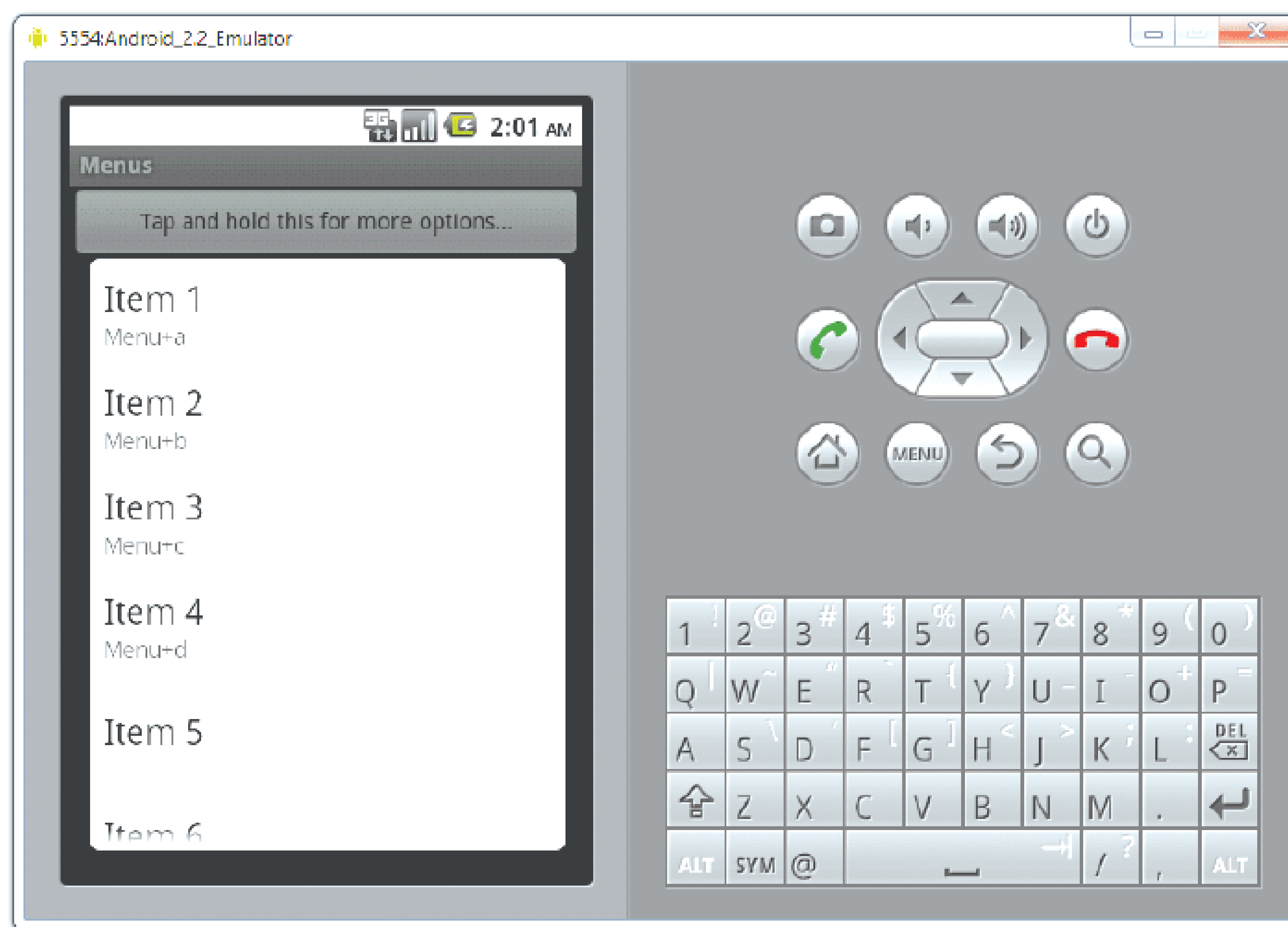


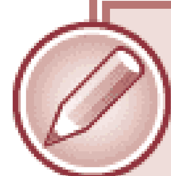
图 5-16

5.3 其他一些视图

除了目前已经了解的标准视图之外，Android SDK还提供了其他一些可使应用程序变得更加有趣的视图。本节中，我们将学习以下视图：AnalogClock、DigitalClock和WebView。

5.3.1 AnalogClock和DigitalClock视图

AnalogClock视图显示一个有两根指针(一根分针，一根时针)的模拟时钟，而DigitalClock视图以数字形式显示时间。它们二者都显示的是系统时间，不允许用户用来显示一个特定时间。因此，如果您打算显示一个特定区域的时间，就必须构建一个自己的定制视图。



注意：在Android中创建一个自己的定制视图已经超出了本书的范围。不过，如果您对这一领域感兴趣，可以在Google的Android文档中查看这一主题：http://developer.android.com/guide/topics/ui/custom-components.html。

使用AnalogClock和DigitalClock视图是很简单的。只需要在XML文件(例如main.xml)中声明它们，如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
```

```
    <AnalogClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <DigitalClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

图5-17展示了运用中的AnalogClock和DigitalClock视图。

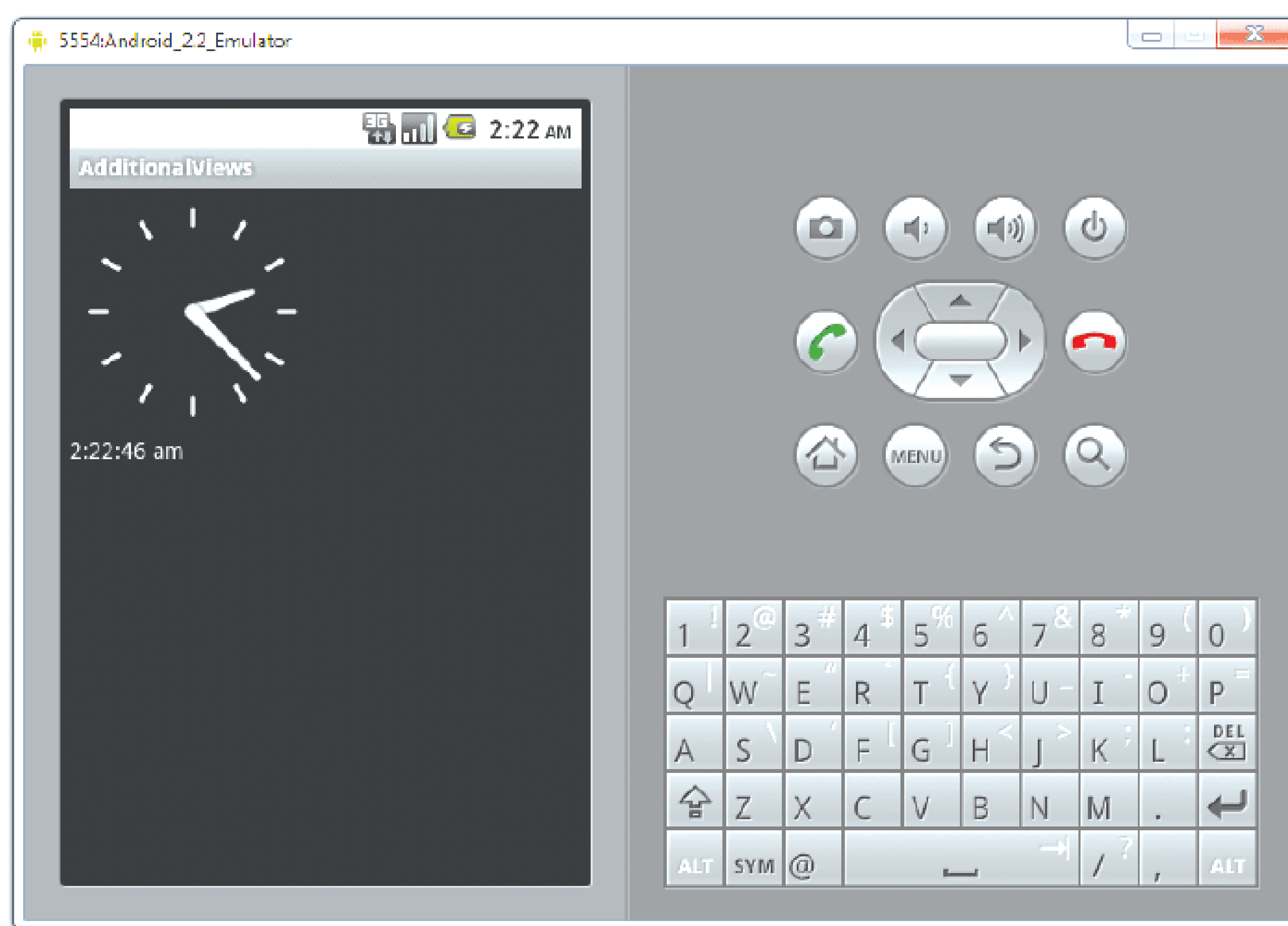


图 5-17

5.3.2　WebView

WebView可以使您在活动中嵌入一个Web浏览器。如果应用程序需要嵌入一些Web内容——如来自其他一些提供商的地图等——这个视图就很有用。下面的“试一试”展示了如何以编程方式加载一个Web页面的内容并在活动中显示出来。

试一试　使用WebView视图

WebView.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个名为WebView的Android新项目。

(2) 在main.xml文件中添加以下语句：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
```

```
    <WebView android:id="@+id/webview1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

(3) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.WebView;

import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        WebView wv = (WebView) findViewById(R.id.webview1);
        WebSettings webSettings = wv.getSettings();
        webSettings.setBuiltInZoomControls(true);
        wv.loadUrl(
        "http://ecx.images-amazon.com/images/I/41HGB-W2Z8L._SL500_AA300_.jpg");
    }
}
```

(4) 按F11键在Android模拟器上调试应用程序。图5-18展示了WebView的内容。



图 5-18

示例说明

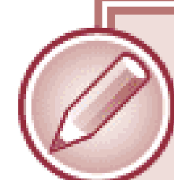
为了利用WebView来加载一个Web页面，可以使用loadUrl()方法，如下所示：

```
wv.loadUrl(
"http://ecx.images-amazon.com/images/I/41HGB-W2Z8L._SL500_AA300_.jpg");
```

要显示内置的缩放控件，需要首先从WebView中获取WebSettings属性，然后调用它的setBuiltInZoomControls()方法：

```
WebSettings webSettings = wv.getSettings();
webSettings.setBuiltInZoomControls(true);
```

图5-19展示了在Android模拟器上使用鼠标单击并拖拽WebView内容时显示的内置缩放控件。



注意：尽管大多数Android设备支持多点触摸屏幕，但当在Android模拟器上测试应用程序时，可以使用内置的缩放控件对Web内容进行缩放。



图 5-19

有时，当加载一个会重定向的页面时(例如加载www.wrox.com会重定向到www.wrox.com/wileyCDA)，WebView将导致应用程序启动设备的浏览器应用程序来加载所需的页面。例如，如果让WebView加载www.wrox.com，Wrox.com将自动重定向到www.wrox.com/WileyCDA/。这时，应用程序将自动启动设备的浏览器应用程序来加载该页面。在图5-20中，注意屏幕顶端的URL栏。

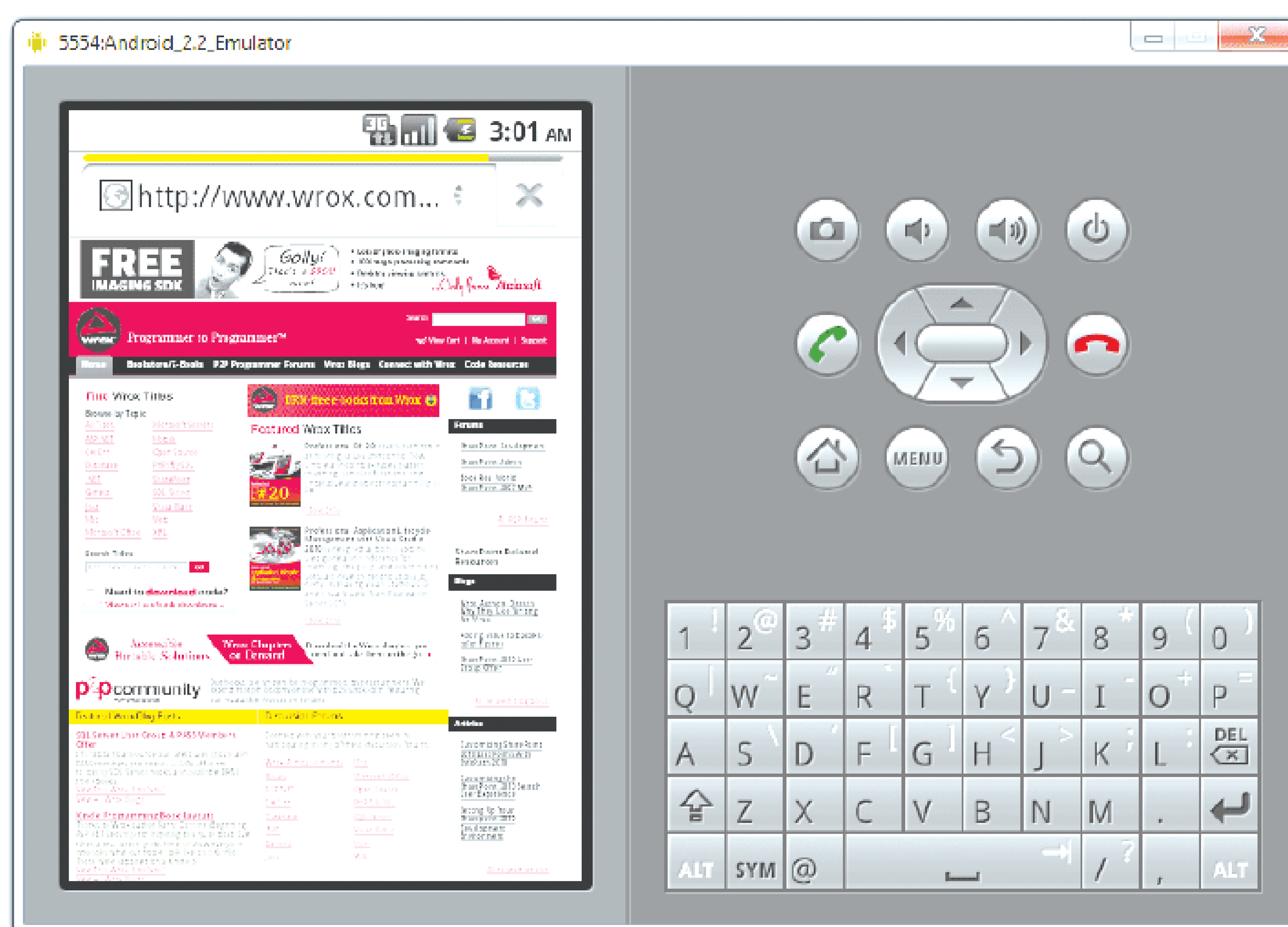


图 5-20

为了避免这种情况发生，需要实现WebViewClient类并重写shouldOverride UrlLoading()方法，如下面的示例所示：

```
package net.learn2develop.WebView;

import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        WebView wv = (WebView) findViewById(R.id.webview1);
        wv.setWebViewClient(new Callback());
        WebSettings webSettings = wv.getSettings();
        webSettings.setBuiltInZoomControls(true);
        wv.loadUrl("http://www.wrox.com");
    }

    private class Callback extends WebViewClient {
        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
            return(false);
        }
    }
}
```

图5-21展示了现在在WebView中正确加载了Wrox.com主页。



图 5-21

还可以使用loadDataWithBaseURL()方法动态创建一个HTML字符串并加载到WebView中：

```
WebView wv = (WebView) findViewById(R.id.webview1);
final String mimeType = "text/html";
final String encoding = "UTF-8";
String html = "<H1>A simple HTML page</H1><body>" +
    "<p>The quick brown fox jumps over the lazy dog</p>";
wv.loadDataWithBaseURL("", html, mimeType, encoding, "");
```

图5-22展示了由WebView所显示的内容。

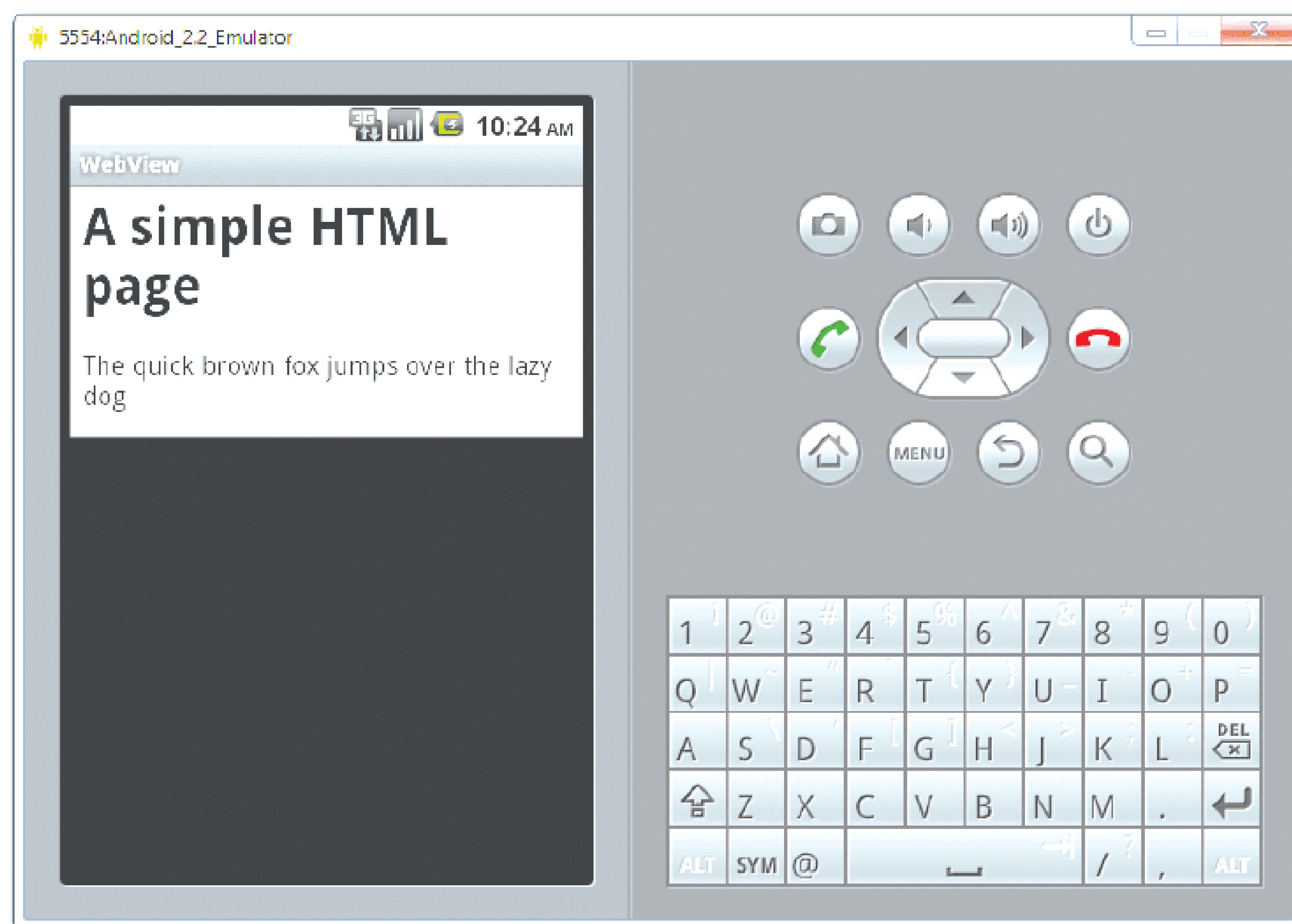


图 5-22

此外，如果在assets文件夹下有一个HTML文件(如图5-23所示)，还可以使用loadUrl()方法将其加载到WebView中：

```
WebView wv = (WebView) findViewById(R.id.webview1);
wv.loadUrl("file:///android_asset/Index.html");
```

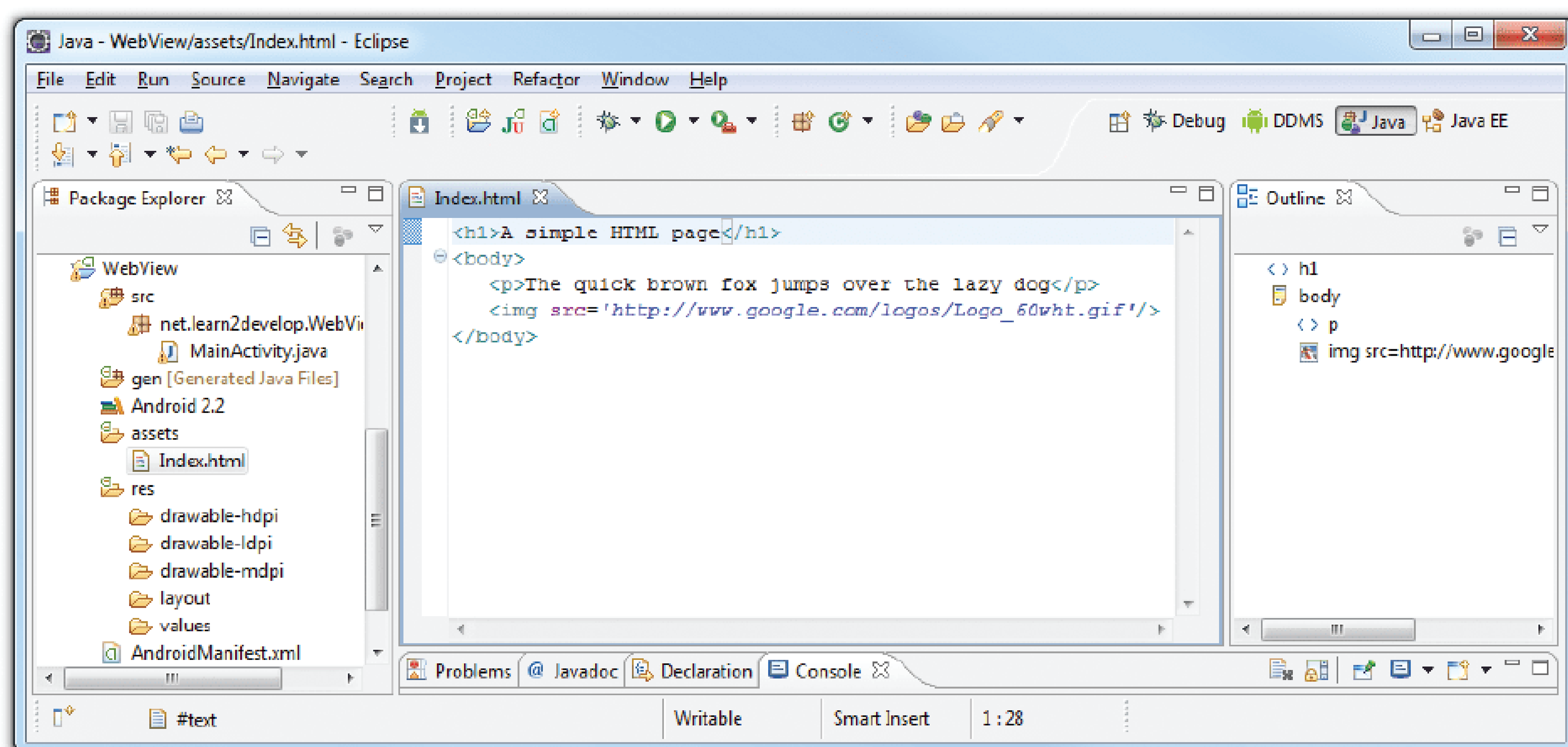


图 5-23

图5-24展示了WebView的内容。



图 5-24

5.4 本章小结

本章中，我们学习了可用来显示图像的各种视图：Gallery、ImageView、ImageSwitcher和GridView。此外，还了解了选项菜单和上下文菜单的不同，以及如何在应用程序中显示它们。最后，我们还学习了AnalogClock和DigitalClock视图(它们以图形化的方式显示当前时间)，以及用来显示Web页面内容的WebView视图。

练　习

1. ImageSwitcher的作用是什么?
2. 说出在活动中实现选项菜单时需要重写的两个方法。
3. 说出在活动中实现上下文菜单时需要重写的两个方法。
4. 当在WebView中发生重定向时如何防止其启动设备的Web浏览器?

练习答案参见附录C。

本章主要内容

主　　题	关 键 概 念
Gallery视图的使用	在水平滚动列表中显示一系列图像
Gallery	<Gallery android:id="@+id/gallery1" android:layout_width="fill_parent" android:layout_height="wrap_content" />
ImageView	<ImageView android:id="@+id/image1" android:layout_width="320px" android:layout_height="250px" android:scaleType="fitXY" />
ImageSwitcher的使用	当在图像间切换时应用动画效果
ImageSwitcher	<ImageSwitcher android:id="@+id/switcher1" android:layout_width="fill_parent" android:layout_height="fill_parent" android:layout_alignParentLeft="true" android:layout_alignParentRight="true" android:layout_alignParentBottom="true" />
GridView的使用	在一个二维的滚动网格中显示项
GridView	<GridView xmlns:android="http://schemas.android.com/apk/res/android" android:id="@+id/gridview" android:layout_width="fill_parent" android:layout_height="fill_parent" android:numColumns="auto_fit" android:verticalSpacing="10dp" android:horizontalSpacing="10dp" android:columnWidth="90dp" android:stretchMode="columnWidth" android:gravity="center" />
AnalogClock	<AnalogClock android:layout_width="wrap_content" android:layout_height="wrap_content" />
DigitalClock	<DigitalClock android:layout_width="wrap_content" android:layout_height="wrap_content" />
WebView	<WebView android:id="@+id/webview1" android:layout_width="wrap_content" android:layout_height="wrap_content" />

第6章 数据持久化

本章将介绍以下内容

- 如何使用SharedPreferences对象保存简单数据
- 如何写入和读取内部和外部存储器中的文件
- 如何创建并使用SQLite数据库

在本章中，我们将学习如何在Android应用程序中保持数据。由于用户希望在稍后阶段中重用数据，因此持久化数据是应用程序开发中的一个重要主题。对于Android来说，持久化数据主要有3个基本的方法：

- 用来保存小块数据的轻量级的机制——共享首选项(shared preferences)
- 传统的文件系统
- 通过SQLite数据库支持的关系数据库管理系统

本章中讨论的技术可以使应用程序创建和访问它们自己的私有数据。在第7章，我们将学习如何跨应用程序共享数据。

6.1 保存和加载用户首选项

Android提供了SharedPreferences对象来帮助我们保存简单的应用程序数据。例如，应用程序可能有一个允许用户指定在应用程序中显示的文本字体大小的选项。这时，应用程序需要记住用户对字体的设定值，以便下次他/她再次使用时，它可以正确设置字体大小。为了达到这一目的，可以有多种选择。将数据保存在一个文件中，但这需要执行一些文件管理例程，诸如向文件写数据、表明从文件中读取了多少字符等。另外，如果有诸如文本尺寸、字体名称、首选的背景色等多条信息需要保存时，写文件的任务就会变得更加繁重。

替代写入文本文件的方法是使用数据库，但无论是从开发人员的角度还是考虑到应用程序运行时的性能，存储简单数据使用数据库都夸张了点。

不过，使用SharedPreferences对象的话，可以通过使用键/值对来保存所需的数据——为需要保存的数据指定一个键，然后其和其值将一起被自动保存到一个XML文件中。

6.1.1 使用getSharedPreferences()方法

为了了解SharedPreferences对象的工作原理，下面的“试一试”演示了可以很容易地将用户数据保存到一个XML文件中，然后通过同一个对象来检索数据。

试一试　使用SharedPreferences对象保存数据

SharedPreferences.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，按图6-1所示创建并命名一个Android项目。

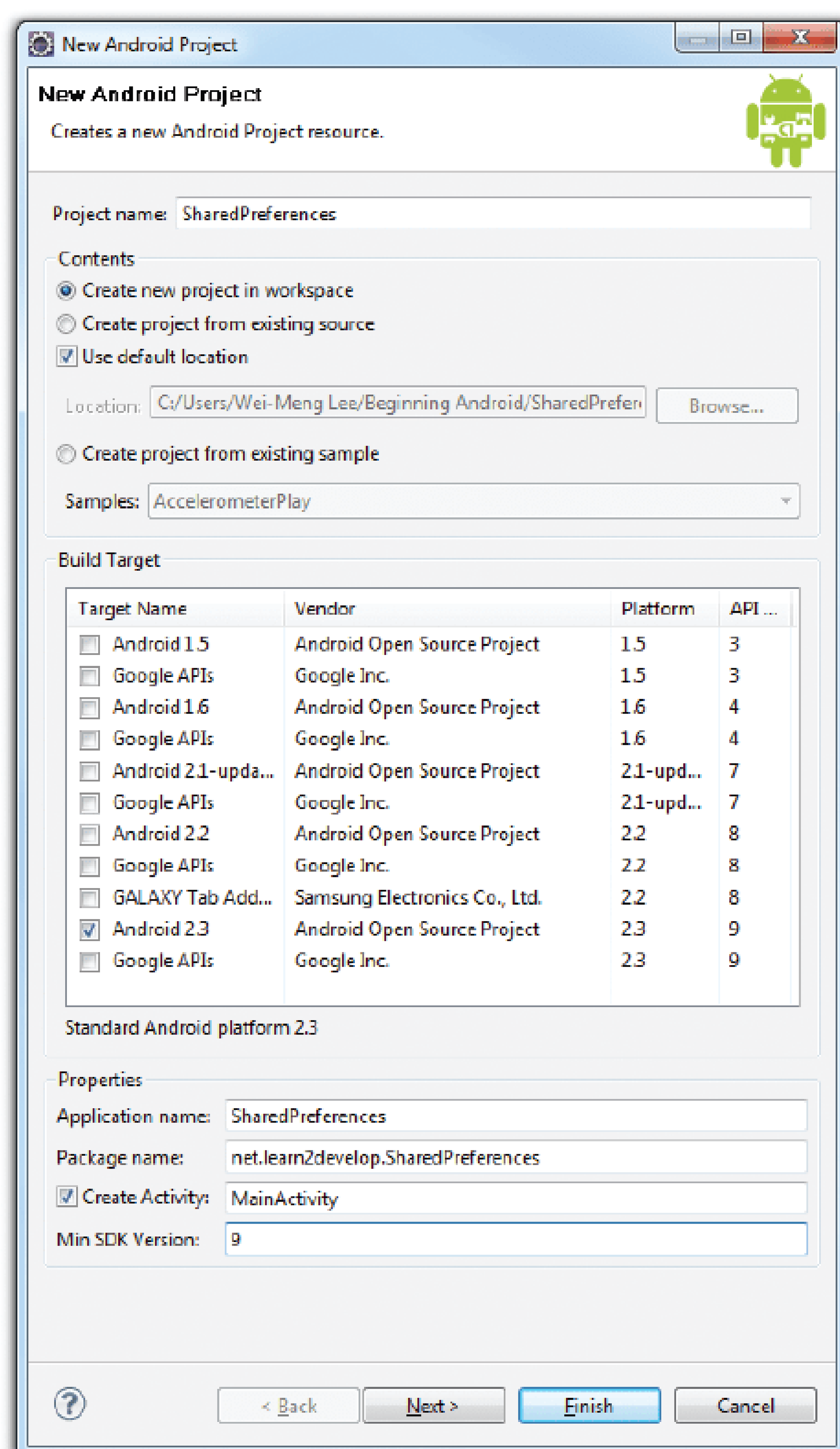


图 6-1

(2) 在main.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<SeekBar
    android:id="@+id/SeekBar01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
<TextView
    android:id="@+id/TextView01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

```
<EditText
    android:id="@+id/EditText01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
<Button
    android:id="@+id/btnSave"
    android:text="Save"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</LinearLayout>
```

(3) 在MainActivity.java文件中添加下列粗体显示的语句:

```
package net.learn2develop.SharedPreferences;

import android.app.Activity;
import android.os.Bundle;

import android.content.SharedPreferences;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.Toast;

public class MainActivity extends Activity {
    private SharedPreferences prefs;
    private String prefName = "MyPref";
    private EditText editText;
    private SeekBar seekBar;
    private Button btn;

    private static final String FONT_SIZE_KEY = "fontsize";
    private static final String TEXT_VALUE_KEY = "textvalue";

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        editText = (EditText) findViewById(R.id.EditText01);
        seekBar = (SeekBar) findViewById(R.id.SeekBar01);
        btn = (Button) findViewById(R.id.btnSave);

        btn.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //---获取SharedPreferences对象---
                prefs = getSharedPreferences(prefName, MODE_PRIVATE);
```

```
                SharedPreferences.Editor editor = prefs.edit();

                //---将EditText视图中的值保存到首选项中---
                editor.putFloat(FONT_SIZE_KEY, editText.getTextSize());
                editor.putString(TEXT_VALUE_KEY, editText.getText().toString());

                //---保存值---
                editor.commit();

                //---显示文件被保存的消息---
                Toast.makeText(getBaseContext(),
                    "Font size saved successfully!",
                    Toast.LENGTH_SHORT).show();
            }
        });

        //---加载SharedPreferences对象---
        SharedPreferences prefs = getSharedPreferences(prefName, MODE_PRIVATE);

        //---将TextView字体大小设置为先前保存的值---
        float fontSize = prefs.getFloat(FONT_SIZE_KEY, 12);

        //---初始化SeekBar和EditText---
        seekBar.setProgress((int) fontSize);
        editText.setText(prefs.getString(TEXT_VALUE_KEY, ""));
        editText.setTextSize(seekBar.getProgress());

        seekBar.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }

            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
                    boolean fromUser) {
                //---改变EditText的字体大小---
                editText.setTextSize(progress);
            }
        });
    }
}
```

(4) 按F11键在Android模拟器上调试应用程序。

(5) 在EditText视图中输入一些文本，然后通过调整SeekBar视图改变其字体大小(如图6-2所示)。单击Save按钮。

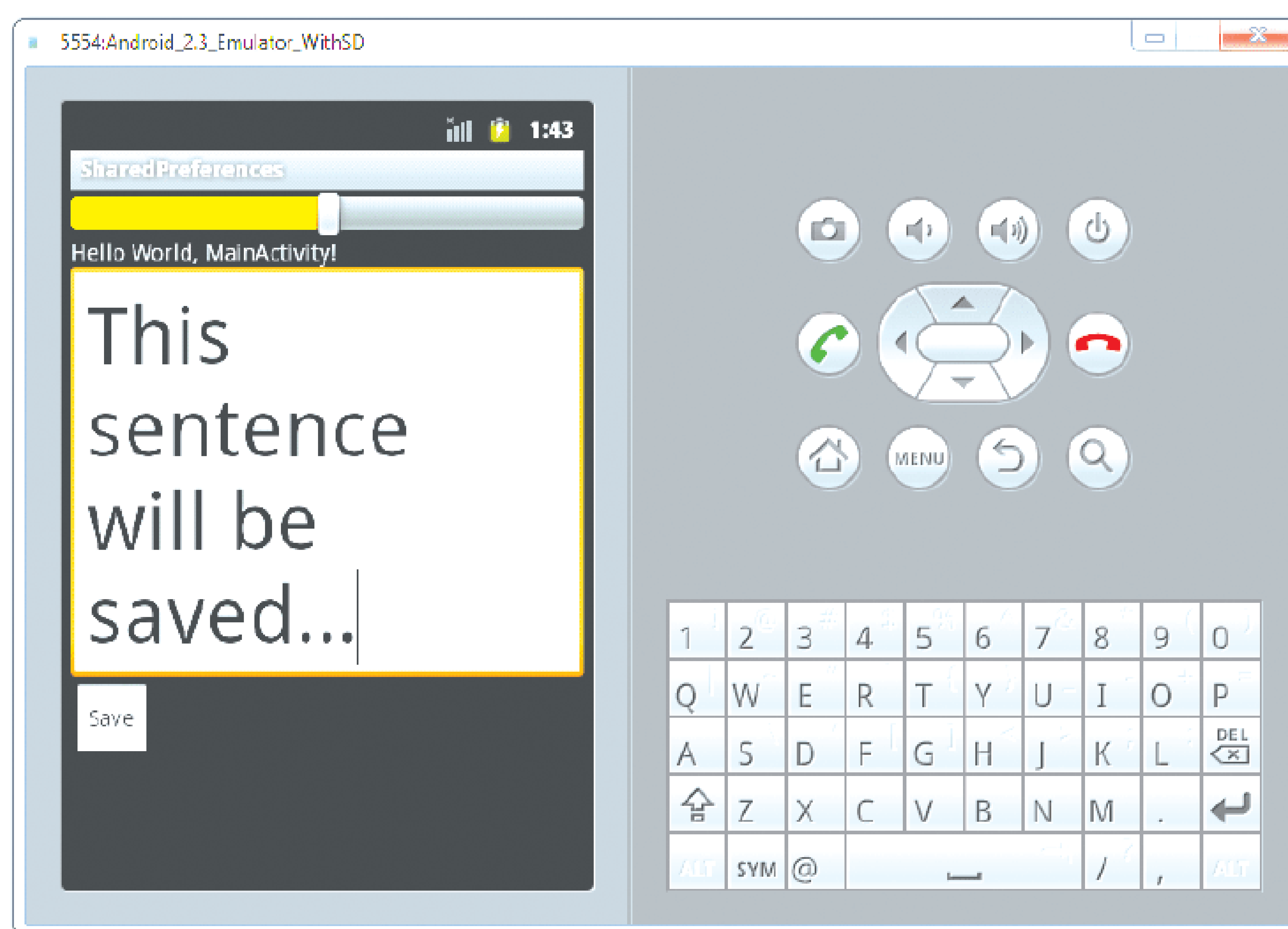


图 6-2

(6) 返回Eclipse，按F11键再次在Android模拟器上调试应用程序。现在，应用程序将显示早先输入的相同的文本内容，字体大小与先前设定的一致。

示例说明

为了使用SharedPreferences对象，要利用getSharedPreferences()方法，给它传递共享首选项文件的名称(所有数据将保存在这一文件中)以及它打开应采用的模式：

```
private SharedPreferences prefs;
...
        //---获取SharedPreferences对象---
        prefs = getSharedPreferences(prefName, MODE_PRIVATE);
        SharedPreferences.Editor editor = prefs.edit();
```

MODE_PRIVATE常量表明共享首选项文件只能由创建它的应用程序打开。Editor类允许利用下列方法将键/值对保存到首选项文件中：

- putString()
- putBoolean()
- putLong()
- putInt()
- putFloat()

当保存完值以后，调用commit()方法来保存修改：

```
        //---将EditText视图中的值保存到首选项中---
        editor.putFloat(FONT_SIZE_KEY, editText.getTextSize());
        editor.putString(TEXT_VALUE_KEY, editText.getText().toString());

        //---保存值---
        editor.commit();
```

当加载活动时，首先获取SharedPreferences对象，然后检索先前保存的所有值：

```
//---加载SharedPreferences对象---
SharedPreferences prefs = getSharedPreferences(prefName, MODE_PRIVATE);

//---将TextView字体大小设置为先前保存的值---
float fontSize = prefs.getFloat(FONT_SIZE_KEY, 12);

//---初始化SeekBar和EditText---
seekBar.setProgress((int) fontSize);
editText.setText(prefs.getString(TEXT_VALUE_KEY, ""));
editText.setTextSize(seekBar.getProgress());
```

共享首选项文件以XML文件形式保存在/data/data/<package_name>/shared_prefs文件夹下(如图6-3所示)。

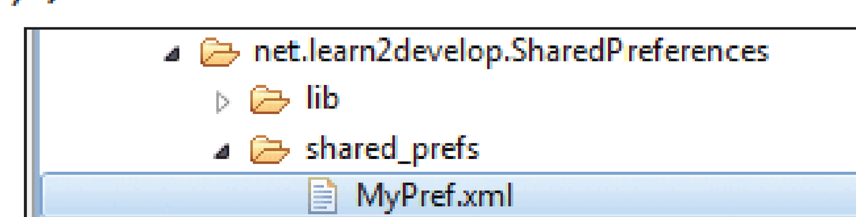


图 6-3

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="textvalue">This is so cool!</string>
    <float name="fontsize" value="75.0" />
</map>
```

6.1.2　使用getPreferences()方法

前面章节中通过给SharedPreferences对象提供一个名称来使用它，如下所示：

```
//---获取SharedPreferences对象---
prefs = getSharedPreferences(prefName, MODE_PRIVATE);
```

在这种情况下，SharedPreferences对象里面保存的信息对于同一应用程序的所有活动都是可见的。不过，如果不需要在活动之间共享数据，可以使用getPreferences()方法，如下所示：

```
//---获取SharedPreferences对象---
prefs = getPreferences(MODE_PRIVATE);
```

getPreferences()方法不需要名称，保存的数据被限制为创建它的活动。在这种情况下，首选项文件的文件名将以创建它的活动命名(如图6-4所示)。

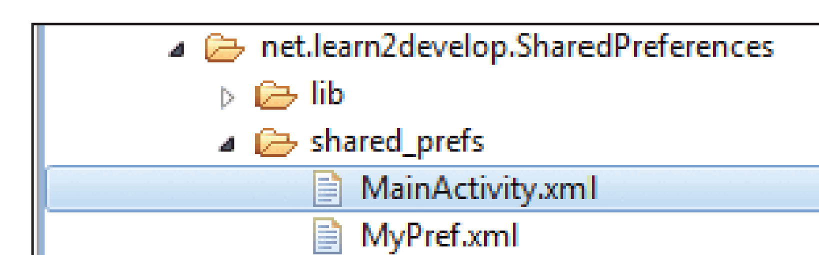


图 6-4

6.2　将数据持久化到文件中

SharedPreferences对象允许您存储最好以键/值对方式进行存储的数据，例如用户ID、生日、性别、驾照号码等数据。然而，有时您还是希望利用传统的文件系统来存储数据。例如，您可能想存储一些诗歌的文本以便在您的应用程序中显示。在Android中，可以使用java.io命名空间中的类来做到这一点。

6.2.1　保存到内部存储器

Android应用程序中保存文件的第一个方法就是将其写入设备的内部存储器。下面的“试一试”展示了如何将用户输入的一个字符串保存到设备的内部存储器中。

试一试 将数据保存到内部存储器

Files.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，按图6-5所示创建并命名一个Android项目。

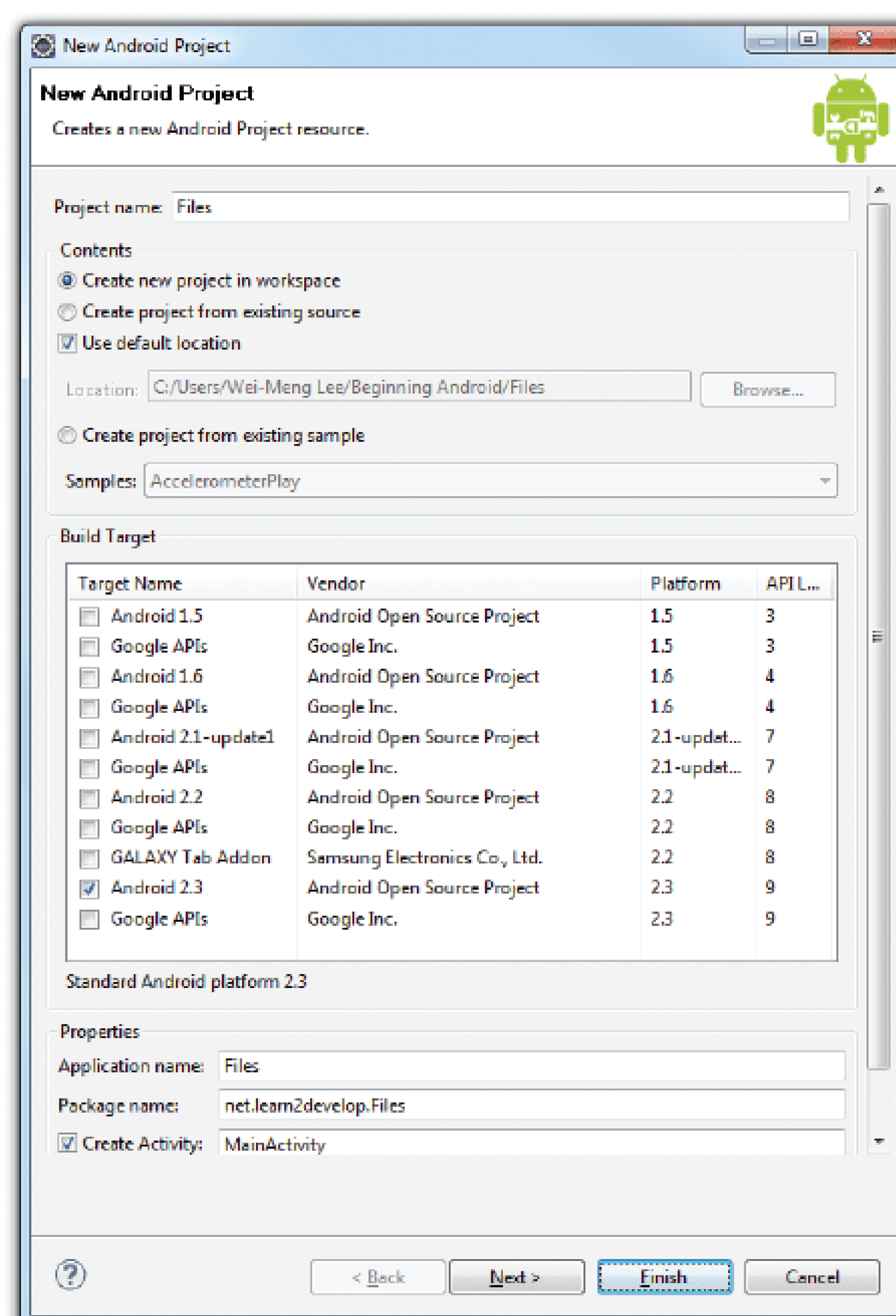


图 6-5

(2) 在main.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Please enter some text"
    />
<EditText
    android:id="@+id/txtText1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
<Button
    android:id="@+id/btnSave"
    android:text="Save"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

```
<Button
    android:id="@+id/btnLoad"
    android:text="Load"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
</LinearLayout>
```

(3) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Files;

import android.app.Activity;
import android.view.View;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity {
    private EditText textBox;
    private static final int READ_BLOCK_SIZE = 100;

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        textBox = (EditText) findViewById(R.id.txtText1);
        Button saveBtn = (Button) findViewById(R.id.btnSave);
        Button loadBtn = (Button) findViewById(R.id.btnLoad);

        saveBtn.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String str = textBox.getText().toString();
                try
                {
                    FileOutputStream fOut =
                        openFileOutput("textfile.txt",
                        MODE_WORLD_READABLE);
                    OutputStreamWriter osw = new
                        OutputStreamWriter(fOut);

                    //---将字符串写入文件---
```

```
                osw.write(str);
                osw.flush();
                osw.close();

                //---显示文件被保存的消息---
                Toast.makeText(getBaseContext(),
                    "File saved successfully!",
                    Toast.LENGTH_SHORT).show();

                //---清空EditText---
                textBox.setText("");
            }
            catch (IOException ioe)
            {
                ioe.printStackTrace();
            }
        }
    });

    loadBtn.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            try
            {
                FileInputStream fIn =
                    openFileInput("textfile.txt");
                InputStreamReader isr = new
                    InputStreamReader(fIn);

                char[] inputBuffer = new char[READ_BLOCK_SIZE];
                String s = "";

                int charRead;
                while ((charRead = isr.read(inputBuffer))>0)
                {
                    //---将字符转换成字符串---
                    String readString =
                        String.copyValueOf(inputBuffer, 0,
                        charRead);
                    s += readString;

                    inputBuffer = new char[READ_BLOCK_SIZE];
                }
                //---将EditText设置为已读取的文本---
                textBox.setText(s);

                Toast.makeText(getBaseContext(),
                    "File loaded successfully!",
                    Toast.LENGTH_SHORT).show();
            }
            catch (IOException ioe) {
```

```
                    ioe.printStackTrace();
                }
            }
        });
    }
}
```

(4) 按F11键在Android模拟器上调试应用程序。

(5) 在EditText视图中输入一些文本(如图6-6所示)，然后单击Save按钮。

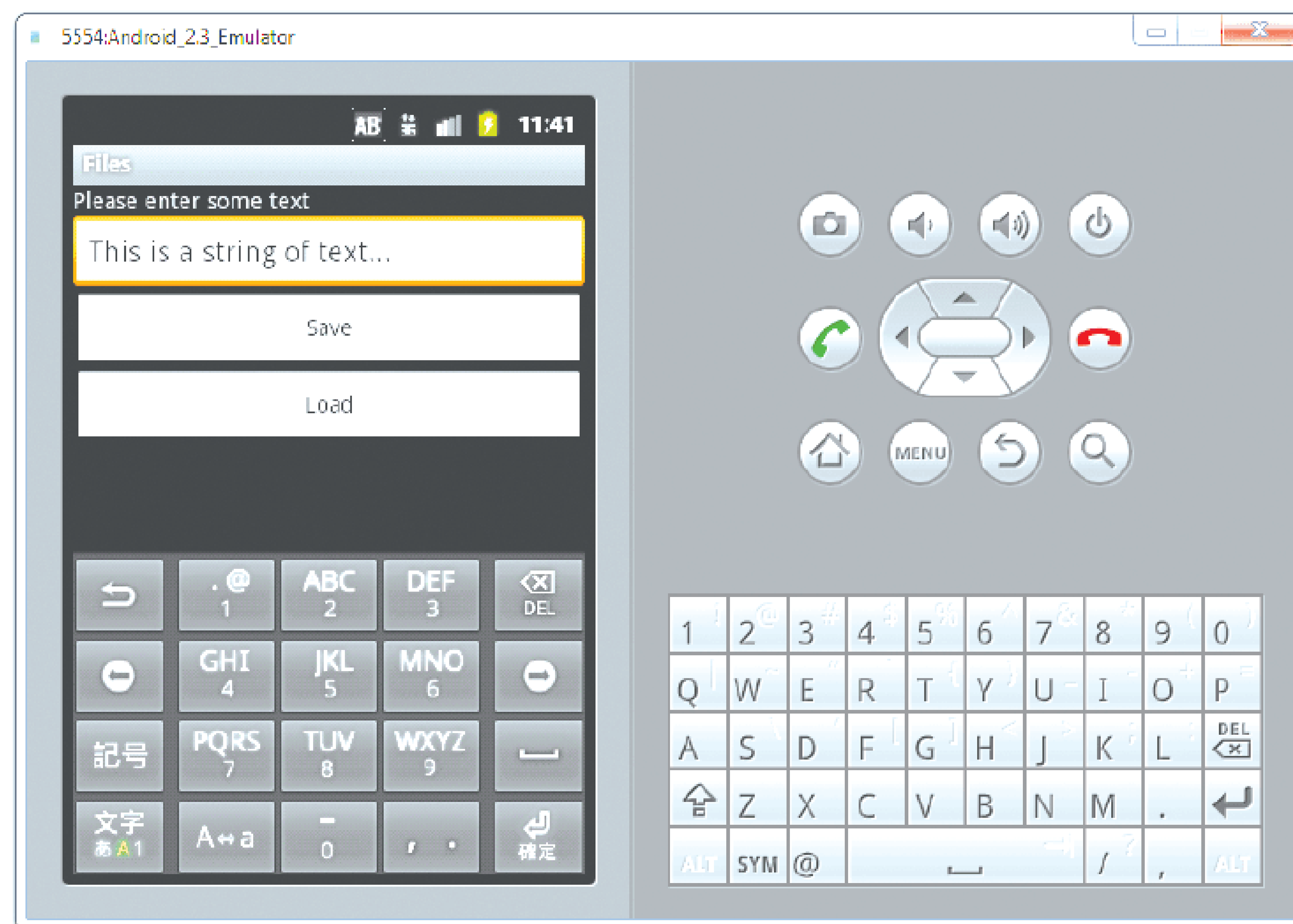


图 6-6

(6) 如果文件成功保存，将看到由Toast类显示出的消息“File saved successfully!”。EditText视图中的文本将消失。

(7) 单击Load按钮，将又会看到字符串出现在EditText视图中。这表明文本被正确地保存了。

示例说明

为了将文本保存到文件中，要使用FileOutputStream类。openFileOutput()方法用指定的模式，打开一个指定的文件来写入。在本例中，使用MODE_WORLD_READABLE常量来表示此文件可被其他所有应用程序读取：

```
FileOutputStream fOut =
    openFileOutput("textfile.txt",
    MODE_WORLD_READABLE);
```

除了MODE_WORLD_READABLE常量之外，还可以从以下模式中进行选择：MODE_PRIVATE(文件只能被创建它的应用程序访问)、MODE_APPEND(附加到现有文件)和MODE_WORLD_WRITEABLE(文件对于其他所有应用程序来说都是可写的)。

为了将字符流转换为字节流，要利用OutputStreamWriter类的一个实例，并给它传递一个FileOutputStream对象的实例：

```
OutputStreamWriter osw = new
    OutputStreamWriter(fOut);
```

然后使用其write()方法将字符串写入到文件中。使用flush()方法来保证所有字节都写入文件。最后，使用close()方法来关闭文件：

```
osw.write(str);
osw.flush();
osw.close();
```

为了读取文件内容，FileInputStream类和InputStreamReader类要配合使用：

```
FileInputStream fIn =
    openFileInput("textfile.txt");
InputStreamReader isr = new
    InputStreamReader(fIn);
```

由于不清楚要读取的文件的大小，因此按100个字符为一块将文件内容读到缓冲区(字符数组)中。然后将所读字符复制到一个String对象中：

```
char[] inputBuffer = new char[READ_BLOCK_SIZE];
String s = "";

int charRead;
while ((charRead = isr.read(inputBuffer))>0)
{
    //---将字符转换为字符串---
    String readString =
        String.copyValueOf(inputBuffer, 0,
        charRead);
    s += readString;

   inputBuffer = new char[READ_BLOCK_SIZE];
}
```

InputStreamReader对象的read()方法读取所读字符个数，如果到达文件末尾就返回-1。

当在Android模拟器上测试此应用程序时，可以使用DDMS来验证应用程序的确将文件保存在了其files目录中(如图6-7所示；实际目录是/data/data/net.learn2develop.Files/files)。

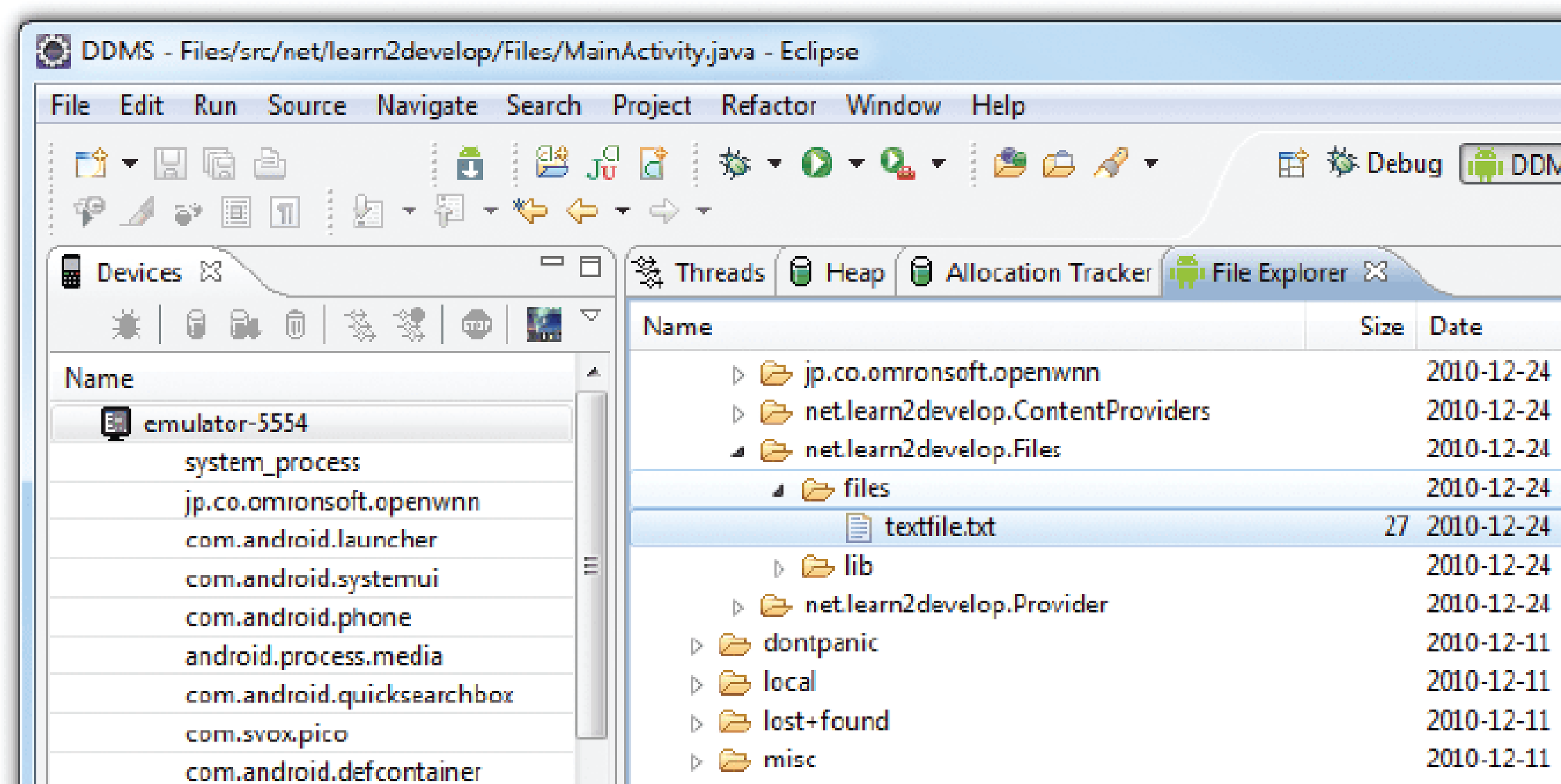


图 6-7

6.2.2 保存到外部存储器(SD卡)

前面一节讲述了如何将文件保存到Android设备的内部存储器。有时，将文件保存到外部存储器也是很有用的，这是因为外部存储器(例如SD卡)容量大，而且很容易和其他用户共享文件(只要将SD卡移到别人的设备里就行了)。

使用前一节中创建的项目作为例子，为了将用户输入的文本保存在SD卡中，按粗体显示内容修改Save按钮的onClick()方法：

```
saveBtn.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {

            String str = textBox.getText().toString();
            try
            {
                //---SD卡存储器---
                File sdCard = Environment.getExternalStorageDirectory();
                File directory = new File (sdCard.getAbsolutePath() +
                    "/MyFiles");
                directory.mkdirs();
                File file = new File(directory, "textfile.txt");
                FileOutputStream fOut = new FileOutputStream(file);

                OutputStreamWriter osw = new
                OutputStreamWriter(fOut);

                //---将字符串写入文件---
                osw.write(str);
                osw.flush();
                osw.close();

                //---显示文件被保存的消息---
                Toast.makeText(getBaseContext(),
                    "File saved successfully!",
                    Toast.LENGTH_SHORT).show();

                //---清空EditText---
                textBox.setText("");
             }
             catch (IOException ioe)
             {
                 ioe.printStackTrace();
             }
        }
    });
```

上述代码使用getExternalStorageDirectory()方法返回外部存储器的完整路径。一般地，对于真实设备返回/sdcard路径，对于Android模拟器返回/mnt/sdcard路径。然而，永远不要试图用硬编码的方式指定SD卡的路径，因为制造商可能会给SD卡指派一个不同的路径名称。因此，应确保使用getExternalStorageDirectory()方法来返回指向SD卡的完整路径。

然后在SD卡中创建一个名为MyFiles的目录。最终，文件将保存在此目录下。

为了从外部存储器中加载文件，需要为Load按钮修改onClick()方法：

```
loadBtn.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            try
            {
                //---SD存储器---
                File sdCard = Environment.getExternalStorageDirectory();
                File directory = new File (sdCard.getAbsolutePath() +
                    "/MyFiles");
                File file = new File(directory, "textfile.txt");
                FileInputStream fIn = new FileInputStream(file);
                InputStreamReader isr = new InputStreamReader(fIn);

                char[] inputBuffer = new char[READ_BLOCK_SIZE];
                String s = "";

                int charRead;
                while ((charRead = isr.read(inputBuffer))>0)
                {
                    //---将字符转换为字符串---
                    String readString =
                        String.copyValueOf(inputBuffer, 0, charRead);
                    s += readString;

                   inputBuffer = new char[READ_BLOCK_SIZE];
                }
                //---将EditText设置为已读取的文本---
                textBox.setText(s);

                Toast.makeText(getBaseContext(),
                    "File loaded successfully!",
                    Toast.LENGTH_SHORT).show();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
    });
```

注意，要写入到外部存储器，需要在AndroidManifest.xml文件中添加WRITE_EXTERNAL_STORAGE权限：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Files"
      android:versionCode="1"
      android:versionName="1.0">
```

```
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE">
    </uses-permission>
</manifest>
```

6.2.3　选择最佳存储选项

目前，您已经了解到几种在Android应用程序中保存数据的方法——SharedPreferences、内部存储器和外部存储器。在应用程序中应该选择哪一种呢？以下是一些建议：

- 如果数据可以用键/值对表示，那么使用SharedPreferences对象。例如，如果想要存储用户首选项的数据，如用户姓名、背景色、生日、最后登录日期等，那么使用SharedPreferences对象来存储这些数据是一个理想的方法。而且，您也不用亲自操心这些数据是如何存储的；所有需要您做的只是使用SharedPreferences对象存储和检索它们。
- 如果需要存储临时数据，使用内部存储器是一个好的选择。例如，应用程序(如一个RSS阅读器)可能需要显示从Web上下载的图像。在这种情况下，将图像保存在内部存储器上是一个好的解决方法。您也许还希望持久化用户所创建的数据，例如有一个备忘录应用程序可以用来让用户记些笔记并保存它们以备后用。在所有这些情况下，使用内部存储器是一个好的选择。
- 有时需要和其他用户共享应用程序数据。例如，您创建了一个Android应用程序来记录用户曾经去过的地点的坐标，并想与其他用户分享这些数据。在这种情况下，可以将文件存储在设备的SD卡上，这样用户在后面可以很容易地将数据转移到其他设备(和计算机)上来使用。

6.2.4　使用静态资源

除了在运行时动态地创建和使用文件之外，还可以在设计时将文件添加到包，以便于在运行时使用它。例如，您可能需要与包捆绑一些帮助文件，以便于用户需要时可以显示一些帮助信息。这时，可以在包下面的res/raw文件夹(需要自己创建)下添加文件。图6-8显示了res/raw文件夹包含了一个名为textfile.txt的文件。

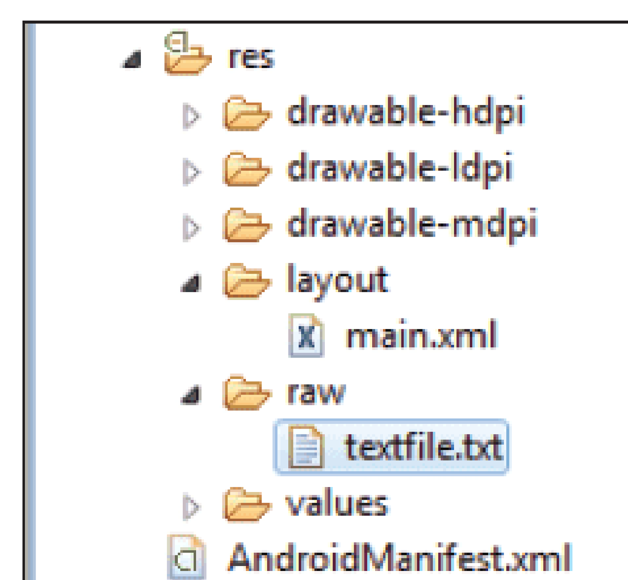


图 6-8

为了在代码中使用此文件，要利用getResources()方法返回一个Resources对象，然后使用其openRawResources()方法来打开包含在res/raw文件夹下的文件：

```
import java.io.InputStream;
import java.io.BufferedReader;
```

```
/** 当活动第一次被创建时调用。 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    InputStream is = this.getResources().openRawResource(R.raw.textfile);
    BufferedReader br = new BufferedReader(new InputStreamReader(is));
    String str = null;
    try {
        while ((str = br.readLine()) != null) {
            Toast.makeText(getBaseContext(),
                str, Toast.LENGTH_SHORT).show();
        }
        is.close();
        br.close();
    } catch (IOException e) {
        e.printStackTrace();
    }

    textBox = (EditText) findViewById(R.id.txtText1);
    Button saveBtn = (Button) findViewById(R.id.btnSave);
    Button loadBtn = (Button) findViewById(R.id.btnLoad);

    saveBtn.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
        }
    });

    loadBtn.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
        }
    });
}
```

存储在res/raw文件夹下的资源的ID是以其文件名来命名的，不带扩展名。例如，如果文本文件是textfile.txt，那么它的资源ID就是R.raw.textfile。

6.3 创建和使用数据库

到目前为止，您所看到的所有技术都用来保存简单的数据集合。对于存储关系型数据，使用数据库会更有效。例如，如果您想存储一所学校所有学生的结果，那么使用数据库来表示它们要有效得多，因为可以使用数据库查询来检索一个特定学生的结果。此外，使用数据库可以通过在不同数据集合间指定关系来强制数据完整性。

Android使用的是SQLite数据库系统。为一个应用程序所创建的数据库只能被此应用程序访问；其他应用程序将不能访问它。

本节中，将学习如何以编程方式在Android应用程序中创建一个SQLite数据库。对于

Android，在一个应用程序中以编程方式创建的SQLite数据库常常存储在/data/data/<package_name>/databases文件夹下。

6.3.1　创建DBAdapter辅助类

处理数据库的一个好的做法是创建一个辅助类来封装访问数据的所有复杂性，使之对于调用它的代码来说是透明的。因此，在本节中将创建一个名为DBAdapter的辅助类，用来创建、打开、关闭和使用一个SQLite数据库。

本例中，将创建一个名为MyDB的数据库，包含一张名为contacts的表。这个表有3列：_id、name和email(如图6-9所示)。

_id	name	email

图 6-9

试一试　创建数据库辅助类

Databases.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个名为Databases的Android项目。

(2) 在项目中新增一个类文件，并命名为DBAdapter.java(如图6-10所示)。

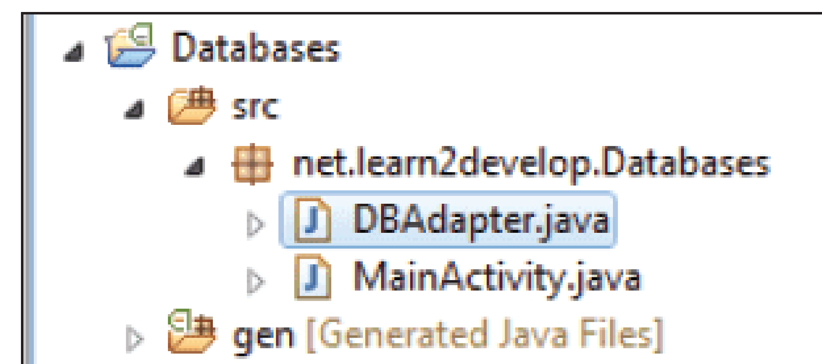


图 6-10

(3) 在DBAdapter.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Databases;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class DBAdapter {
    public static final String KEY_ROWID = "_id";
    public static final String KEY_NAME = "name";
    public static final String KEY_EMAIL = "email";
    private static final String TAG = "DBAdapter";

    private static final String DATABASE_NAME = "MyDB";
    private static final String DATABASE_TABLE = "contacts";
    private static final int DATABASE_VERSION = 1;

    private static final String DATABASE_CREATE =
        "create table contacts (_id integer primary key autoincrement, "
        + "name text not null, email text not null);";

    private final Context context;
```

```
    private DatabaseHelper DBHelper;
    private SQLiteDatabase db;

    public DBAdapter(Context ctx)
    {
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }

    private static class DatabaseHelper extends SQLiteOpenHelper
    {
        DatabaseHelper(Context context)
        {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        @Override
        public void onCreate(SQLiteDatabase db)
        {
            try {
                db.execSQL(DATABASE_CREATE);
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
        {
            Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
                    + newVersion + ", which will destroy all old data");
            db.execSQL("DROP TABLE IF EXISTS contacts");
            onCreate(db);
        }
    }

    //---打开数据库---
    public DBAdapter open() throws SQLException
    {
        db = DBHelper.getWritableDatabase();
        return this;
    }

    //---关闭数据库---
    public void close()
    {
        DBHelper.close();
    }

    //---在数据库中插入一个联系人---
```

```
        public long insertContact(String name, String email)
        {
            ContentValues initialValues = new ContentValues();
            initialValues.put(KEY_NAME, name);
            initialValues.put(KEY_EMAIL, email);
            return db.insert(DATABASE_TABLE, null, initialValues);
        }

        //---删除一个特定的联系人---
        public boolean deleteContact(long rowId)
        {
            return db.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
        }

        //---检索所有的联系人---
        public Cursor getAllContacts()
        {
            return db.query(DATABASE_TABLE, new String[] {KEY_ROWID, KEY_NAME,
                    KEY_EMAIL}, null, null, null, null, null);
        }

        //---检索一个特定的联系人---
        public Cursor getContact(long rowId) throws SQLException
        {
            Cursor mCursor =
                    db.query(true, DATABASE_TABLE, new String[] {KEY_ROWID,
                    KEY_NAME, KEY_EMAIL}, KEY_ROWID + "=" + rowId, null,
                    null, null, null, null);
            if (mCursor != null) {
                mCursor.moveToFirst();
            }
            return mCursor;
        }

        //---更新一个联系人---
        public boolean updateContact(long rowId, String name, String email)
        {
            ContentValues args = new ContentValues();
            args.put(KEY_NAME, name);
            args.put(KEY_EMAIL, email);
              return db.update(DATABASE_TABLE, args, KEY_ROWID + "=" + rowId,
    null) > 0;
        }
    }
```

示例说明

首先为在数据库中将要创建的表定义几个常量来包含不同的字段：

```
        public static final String KEY_ROWID = "_id";
        public static final String KEY_NAME = "name";
```

```
    public static final String KEY_EMAIL = "email";
    private static final String TAG = "DBAdapter";

    private static final String DATABASE_NAME = "MyDB";
    private static final String DATABASE_TABLE = "contacts";
    private static final int DATABASE_VERSION = 1;

    private static final String DATABASE_CREATE =
        "create table contacts (_id integer primary key autoincrement, "
        + "name text not null, email text not null);";
```

特别是，DATABASE_CREATE常量包含了用于在MyDB数据库中创建contacts表的SQL语句。

在DBAdapter类中，还扩展了SQLiteOpenHelper类，它是一个在Android中用来处理数据库创建和版本管理的辅助类。尤其是，它重写了onCreate()方法和onUpgrade()方法：

```
public class DBAdapter {
    public static final String KEY_ROWID = "_id";
    public static final String KEY_NAME = "name";
    public static final String KEY_EMAIL = "email";
    private static final String TAG = "DBAdapter";

    private static final String DATABASE_NAME = "MyDB";
    private static final String DATABASE_TABLE = "contacts";
    private static final int DATABASE_VERSION = 1;

    private static final String DATABASE_CREATE =
        "create table contacts (_id integer primary key autoincrement, "
        + "name text not null, email text not null);";

    private final Context context;

    private DatabaseHelper DBHelper;
    private SQLiteDatabase db;

    public DBAdapter(Context ctx)
    {
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }

    private static class DatabaseHelper extends SQLiteOpenHelper
    {
        DatabaseHelper(Context context)
        {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        @Override
        public void onCreate(SQLiteDatabase db)
```

```
        {
            try {
                db.execSQL(DATABASE_CREATE);
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
        {
            Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
                    + newVersion + ", which will destroy all old data");
            db.execSQL("DROP TABLE IF EXISTS contacts");
            onCreate(db);
        }
    }
```

onCreate()方法在所需数据库不存在时创建一个新的数据库。onUpgrade()方法在数据库需要升级时调用。这可以通过检查在DATABASE_VERSION常量中定义的值来实现。对于onUpgrade()方法的这一实现，只是删除表并再创建它。

然后，可以定义用于打开和关闭数据库的不同方法，以及用于在表中添加、修改、删除行的方法。

```
public class DBAdapter {
//...
//...

    //---打开数据库---
    public DBAdapter open() throws SQLException
    {
        db = DBHelper.getWritableDatabase();
        return this;
    }

    //---关闭数据库---
    public void close()
    {
        DBHelper.close();
    }

    //---在数据库中插入一个联系人---
    public long insertContact(String name, String email)
    {
        ContentValues initialValues = new ContentValues();
        initialValues.put(KEY_NAME, name);
        initialValues.put(KEY_EMAIL, email);
        return db.insert(DATABASE_TABLE, null, initialValues);
    }
```

```
        //---删除一个特定的联系人---
        public boolean deleteContact(long rowId)
        {
            return db.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
        }

        //---检索所有的联系人---
        public Cursor getAllContacts()
        {
            return db.query(DATABASE_TABLE, new String[] {KEY_ROWID, KEY_NAME,
                    KEY_EMAIL}, null, null, null, null, null);
        }

        //---检索一个特定的联系人---
        public Cursor getContact(long rowId) throws SQLException
        {
            Cursor mCursor =
                    db.query(true, DATABASE_TABLE, new String[] {KEY_ROWID,
                    KEY_NAME, KEY_EMAIL}, KEY_ROWID + "=" + rowId, null,
                    null, null, null, null);
            if (mCursor != null) {
                mCursor.moveToFirst();
            }
            return mCursor;
        }

        //---更新一个联系人---
        public boolean updateContact(long rowId, String name, String email)
        {
            ContentValues args = new ContentValues();
            args.put(KEY_NAME, name);
            args.put(KEY_EMAIL, email);
              return db.update(DATABASE_TABLE, args, KEY_ROWID + "=" + rowId,
    null) > 0;
        }
    }
```

注意，Android使用Cursor类作为查询的返回值。可以将Cursor看成是一个指向数据库查询的结果集的指针。使用Cursor可以使Android更有效地按需要管理行和列。

使用ContentValues对象来存储键/值对。其put()方法可用于插入具有不同数据类型值的键。

6.3.2 以编程方式使用数据库

接下来将利用前一节创建的辅助类来使用数据库。

1. 添加联系人

下面的“试一试”演示了如何将一个联系人添加到表中。

试一试　向表中添加联系人

(1) 打开先前创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Databases;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        DBAdapter db = new DBAdapter(this);

        //---添加一个联系人---
        db.open();
        long id = db.insertContact("Wei-Meng Lee", "weimenglee@
        learn2develop.net");
        id = db.insertContact("Mary Jackson", "mary@jackson.com");
        db.close();
    }
}
```

(2) 按F11键在Android模拟器上调试应用程序。

示例说明

在这个例子中，首先创建了一个DBAdapter类的实例：

```
        DBAdapter db = new DBAdapter(this);
```

insertContact()方法返回所插入行的ID。如果在操作中发生错误则返回−1。

如果使用DDMS检查Android设备/模拟器的文件系统，可以看到在databases文件夹下创建了MyDB数据库(如图6-11所示)。

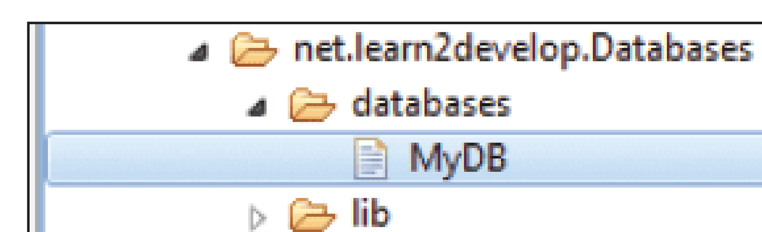


图 6-11

2. 检索所有联系人

为了在contacts表中检索所有联系人，使用DBAdapter类的getAllContacts()方法，如下面的“试一试”所示。

试一试　从表中检索所有联系人

(1) 打开先前创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Databases;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Toast;

import android.database.Cursor;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        DBAdapter db = new DBAdapter(this);

        /*
        //---添加一个联系人---
        db.open();
        long id = db.insertContact("Wei-Meng Lee", "weimenglee@
        learn2develop.net");
        id = db.insertContact("Mary Jackson", "mary@jackson.com");
        db.close();
        */

        //---获取所有联系人---
        db.open();
        Cursor c = db.getAllContacts();
        if (c.moveToFirst())
        {
            do {
                DisplayContact(c);
            } while (c.moveToNext());
        }
        db.close();
    }

    public void DisplayContact(Cursor c)
    {
        Toast.makeText(this,
                "id: " + c.getString(0) + "\n" +
                "Name: " + c.getString(1) + "\n" +
                "Email:  " + c.getString(2),
                Toast.LENGTH_LONG).show();
    }
}
```

(2) 按F11键在Android模拟器上调试应用程序。图6-12展示了Toast类所显示的从数据库中检索到的联系人。

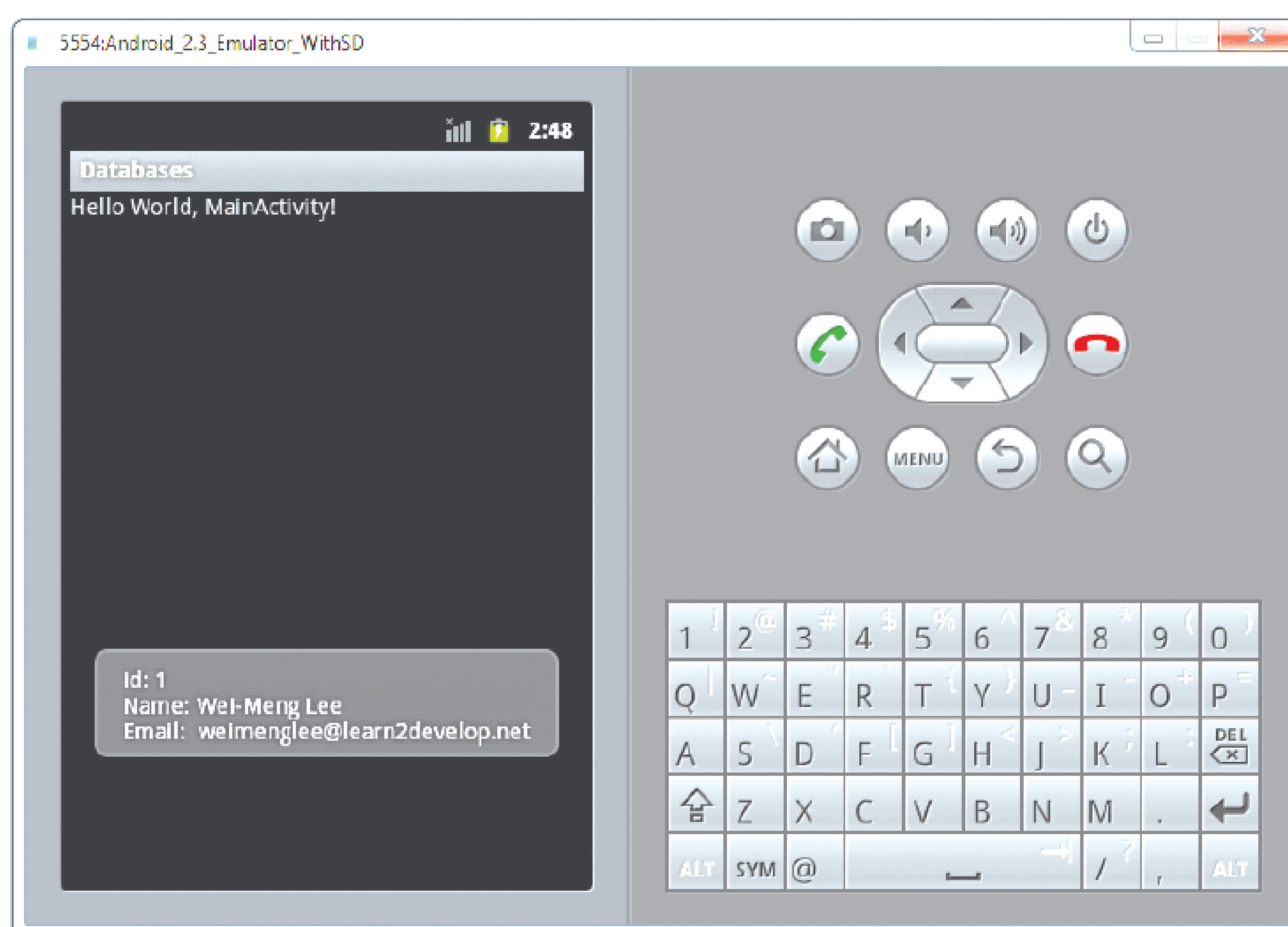


图 6-12

示例说明

DBAdapter类的getAllContacts()方法检索存储于数据库中的所有联系人。结果以Cursor对象形式返回。为了显示所有联系人，首先需要调用Cursor对象的moveToFirst()方法。如果成功(这意味着至少有一行可用)，则使用DisplayContact()方法来显示联系人的详细信息。要移动到下一个联系人，则要调用Cursor对象的moveToNext()方法。

3. 检索单个联系人

要利用联系人的ID来检索单个联系人，可按下面的“试一试”所展示的，调用DBAdapter类的getContact()方法。

试一试　从表中检索单个联系人

(1) 打开先前创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    DBAdapter db = new DBAdapter(this);

    /*
    //---添加一个联系人---
    //...
    */

    /*
    //---获取所有联系人---
```

```
        //...
        */

        //---获取一个联系人---
        db.open();
        Cursor c = db.getContact(2);
        if (c.moveToFirst())
            DisplayContact(c);
        else
            Toast.makeText(this, "No contact found", Toast.LENGTH_LONG).
            show();
        db.close();
    }
```

(2) 按F11键在Android模拟器上调试应用程序。第二个联系人的详细信息将通过Toast类显示出来。

示例说明

DBAdapter类的getContact()方法使用联系人的ID来检索单个联系人。传入联系人的ID；这里传入的ID是2，表明想要检索第二个联系人：

```
        Cursor c = db.getContact(2);
```

结果以Cursor对象的形式返回。如果返回了一行，就使用DisplayContact()方法来显示其详细信息；否则使用Toast类显示一条消息。

---

4. 更新联系人

通过调用DBAdapter类的updateContact()方法并传递一个想要更新的联系人的ID，可以实现对某个特定联系人的更新操作，如下面的“试一试”所示。

试一试　更新表中某个联系人

(1) 打开先前创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        DBAdapter db = new DBAdapter(this);

        /*
        //---添加一个联系人---
       //...
        */

        /*
```

```
        //---获取所有联系人---
        //...
        */

        /*
        //...
       */

        //---更新联系人---
        db.open();
        if (db.updateContact(1, "Wei-Meng Lee", "weimenglee@gmail.com"))
            Toast.makeText(this, "Update successful.", Toast.LENGTH_LONG).
            show();
        else
            Toast.makeText(this, "Update failed.", Toast.LENGTH_LONG).
            show();
        db.close();
    }
```

(2) 按F11键在Android模拟器上调试应用程序。如果更新成功，将显示一条消息。

示例说明

DBAdapter类中的updateContact()方法利用您打算更新的联系人的ID来更新此联系人的详细信息。它返回一个Boolean值，表明更新是否成功。

---

5. 删除一个联系人

使用DBAdapter类的deleteContact()方法并传递一个想要删除的联系人的ID，可以实现对某个联系人的删除操作，如下面的“试一试”所示。

试一试 删除表中某个联系人

(1) 打开先前创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        DBAdapter db = new DBAdapter(this);

        /*
        //---添加一个联系人---
       //...
        */

        /*
        //---获取所有联系人---
```

```
//...
 */

 /*
 //---获取一个联系人---
//...
 */

 /*
 //---更新联系人---
//...
 */

 //---删除一个联系人---
 db.open();
 if (db.deleteContact(1))
     Toast.makeText(this, "Delete successful.", Toast.LENGTH_LONG).
     show();
 else
     Toast.makeText(this, "Delete failed.", Toast.LENGTH_LONG).
     show();
 db.close();
}
```

(2) 按F11键在Android模拟器上调试应用程序。如果删除成功，将显示一条消息。

示例说明

DBAdapter类中的deleteContact()方法利用您打算更新的联系人的ID来删除此联系人。它返回一个Boolean值，表明删除是否成功。

6. 升级数据库

有时，在创建和开始使用数据库之后，您可能需要添加另外的表、改变数据库的模式，或者在表中添加一些列。这时，就需要将旧数据库中现有的数据迁移到一个新数据库中。

要升级数据库，需要将DATABASE_VERSION常量改为比先前要高的值。例如，如果先前的值是1，则现在改为2：

```
public class DBAdapter {
    public static final String KEY_ROWID = "_id";
    public static final String KEY_NAME = "name";
    public static final String KEY_EMAIL = "email";
    private static final String TAG = "DBAdapter";

    private static final String DATABASE_NAME = "MyDB";
    private static final String DATABASE_TABLE = "contacts";
    private static final int DATABASE_VERSION = 2;
```

当再一次运行应用程序时，在Eclipse的LogCat窗口中可以看到以下消息：

```
DBAdapter(24096): Upgrading database from version 1 to 2, which will
destroy all old data
```

在这一例子中，为简单起见，直接删除现有的表并创建一个新表。在实际中，通常需要备份现有的表，然后将其内容复制到新表中。

6.3.3　预创建数据库

在实际的应用中，有时在设计时预创建数据库要比在运行时创建更有效。要预创建一个SQLite数据库，可以使用Internet上很多可用的免费工具。其中一个这样的工具就是SQLite Database Browser，其对于不同的平台都是免费可用的(http://sourceforge.net/projects/sqlitebrowser/)。

一旦安装了SQLite Database Browser，就可以用可视化方式来创建一个数据库。图6-13展示了已经创建好一个指明了字段的contacts表。

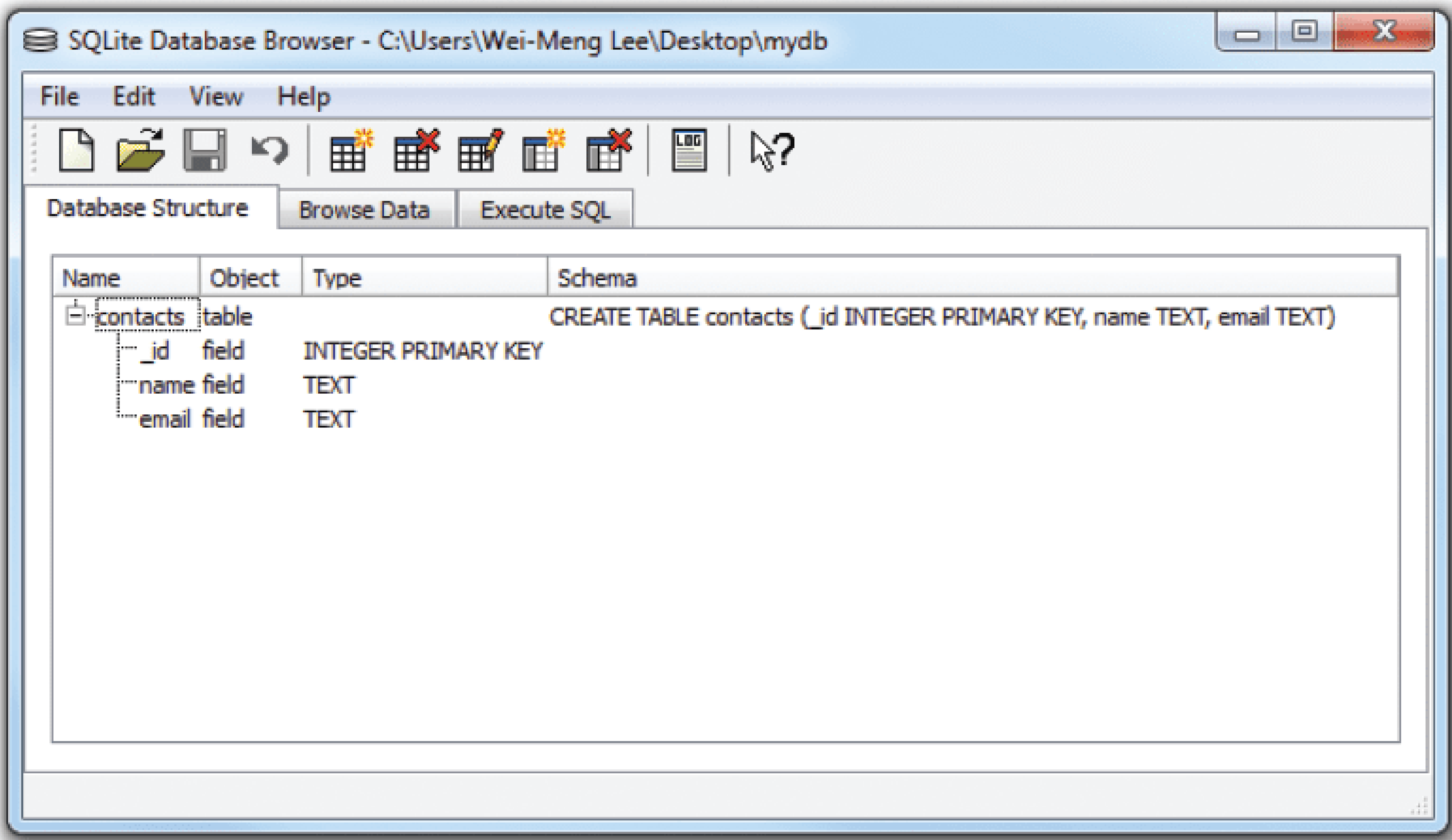


图 6-13

用行填充表也很简单。图6-14展示了如何使用Browse Data选项卡来为表填充数据。

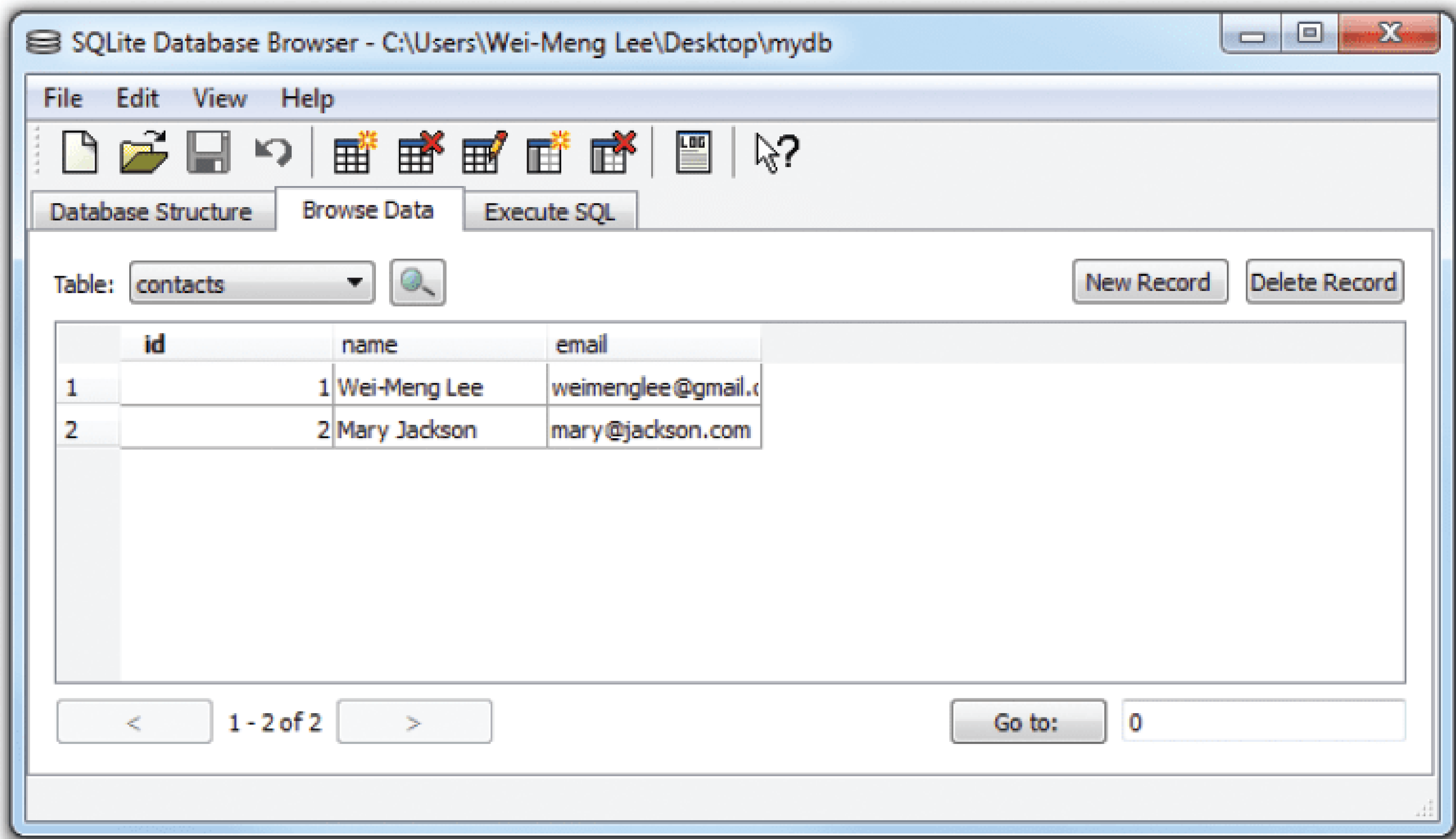


图 6-14

将数据库与应用程序捆绑

在设计时创建了数据库后，下一步要做的就是将其与应用程序捆绑，这样就可以在应用程序中使用数据库了。下面的“试一试”展示了如何捆绑。

试一试　捆绑一个数据库

(1) 打开先前创建的同一个项目，将在前一节中创建的SQLite数据库文件拖放到Eclipse中Android项目的assets文件夹下(如图6-15所示)。

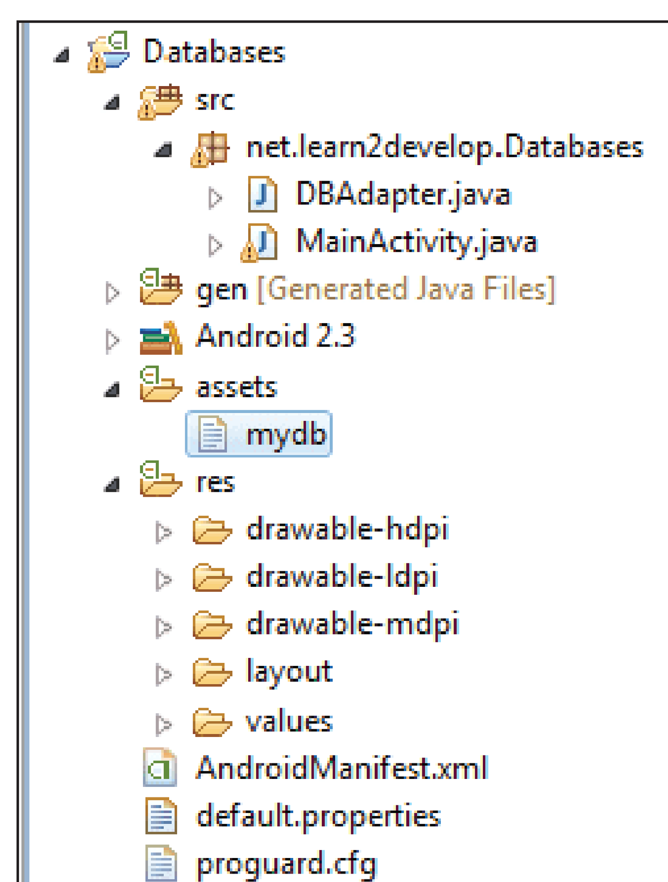
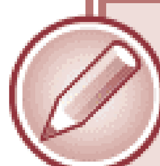


图 6-15



注意：添加到assets文件夹下的文件名必须采用小写字母格式。因此，像MyDB这样的文件名是无效的，而mydb是可以的。

(2) 在MainActivity.java文件中添加下列粗体显示的语句：

```
@Override
   public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.main);
       try {
           String destPath = "/data/data/" + getPackageName() +
               "/databases/MyDB";
           File f = new File(destPath);
           if (!f.exists()) {
               CopyDB( getBaseContext().getAssets().open("mydb"),
                   new FileOutputStream(destPath));
           }
       } catch (FileNotFoundException e) {
           e.printStackTrace();
       } catch (IOException e) {
           e.printStackTrace();
       }

       DBAdapter db = new DBAdapter(this);
```

```
        //---获取所有联系人---
        db.open();
        Cursor c = db.getAllContacts();
        if (c.moveToFirst())
        {
            do {
                DisplayContact(c);
            } while (c.moveToNext());
        }
        db.close();
    }

    public void CopyDB(InputStream inputStream,
    OutputStream outputStream)
    throws IOException {
        //---一次复制1K字节---
        byte[] buffer = new byte[1024];
        int length;
        while ((length = inputStream.read(buffer)) > 0) {
            outputStream.write(buffer, 0, length);
        }
        inputStream.close();
        outputStream.close();
    }
```

(3) 按F11键在Android模拟器上调试应用程序。当应用程序运行时，它将把mydb数据库文件以MyDB为名复制到/data/data/net.learn2develop.Databases/databases/文件夹下。

示例说明

首先将CopyDB()方法定义为将数据库文件从一处复制到另一处：

```
    public void CopyDB(InputStream inputStream,
    OutputStream outputStream)
    throws IOException {
        //---一次复制1K字节---
        byte[] buffer = new byte[1024];
        int length;
        while ((length = inputStream.read(buffer)) > 0) {
            outputStream.write(buffer, 0, length);
        }
        inputStream.close();
        outputStream.close();
    }
```

注意，在这种情况下，使用InputStream对象来从源文件中读取数据，然后使用OutputStream对象将其写入到目标文件中。

当活动被创建时，将位于assets文件夹下的数据库文件复制到Android设备(或模拟器)上的/data/data/net.learn2develop.Databases/databases/文件夹下：

```
try {
    String destPath = "/data/data/" + getPackageName() +
        "/databases/MyDB";
    File f = new File(destPath);
    if (!f.exists()) {
        CopyDB( getBaseContext().getAssets().open("mydb"),
            new FileOutputStream(destPath));
    }
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

只有在目标文件夹下不存在数据库文件时才执行复制操作。如果不进行这一检查，每一次创建活动时，数据库文件将被assets文件夹下的文件所覆盖。这也许是您所不希望的，因为应用程序在运行时很可能会对数据库文件做修改，这将消除到目前为止您对数据库所做出的一切改变。

为了确保数据库文件的确被复制了，在测试应用程序之前要确保在模拟器中删除了数据库文件(如果还存在的话)。可以使用DDMS来删除数据库(如图6-16所示)。

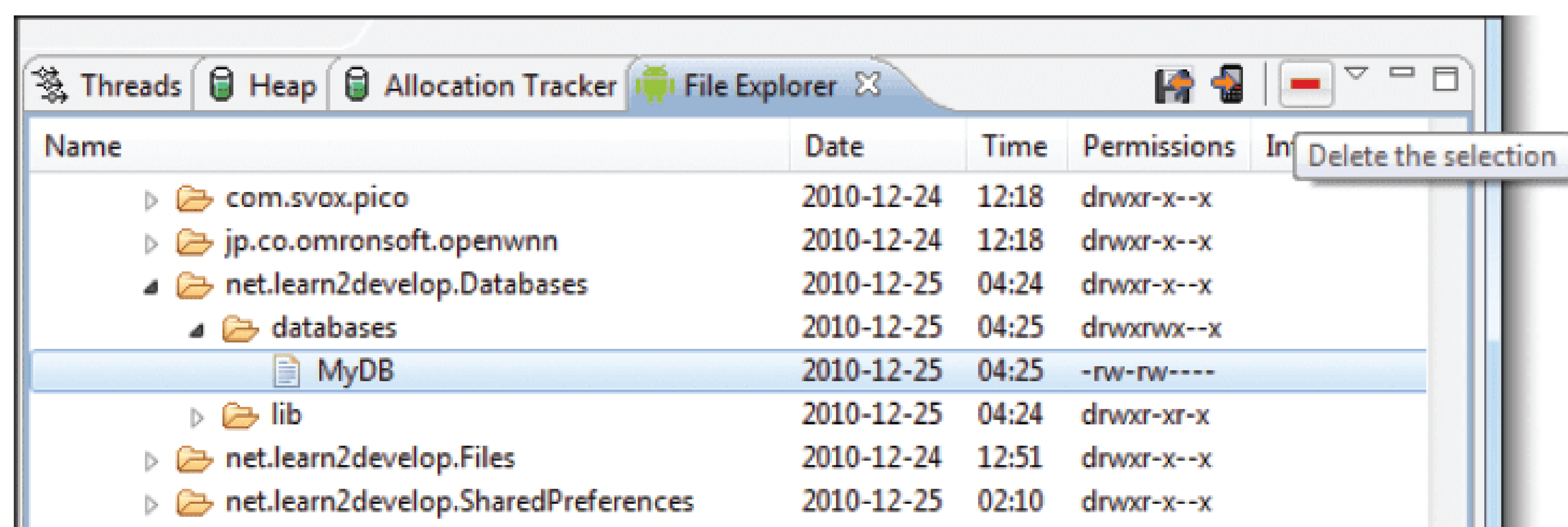


图 6-16

6.4 本章小结

在本章中，我们学习了将持久性数据保存在Android设备中的不同方法。对于简单的非结构化数据，使用SharedPreferences对象是一个理想的方案。如果需要存储批量数据，可以考虑使用传统的文件系统。最后，对于结构化数据，在关系数据库管理系统中进行存储会更有效率。为此，Android提供了可以利用公开的API轻松访问的SQLite数据库。

注意，对于SharedPreferences对象和SQLite数据库来说，数据只能被创建它的应用程序访问。换句话说，它是不可共享的。如果需要在不同应用程序之间共享数据，则要创建一个内容提供者(content provider)。第7章将详细讨论内容提供者。

练 习

1. getSharedPreferences()和getPreferences()方法的区别是什么？

2. 说出能够获取Android设备的外部存储器的路径的方法。

3. 当向外部存储器写入文件时需要声明什么权限?

练习答案参见附录C。

本章主要内容

主　　题	关 键 概 念
保存简单的用户数据	使用SharedPreferences对象
在同一应用程序的活动之间共享数据	使用getSharedPreferences()方法
保存只对创建它的活动可见的数据	使用getPreferences()方法
保存到文件	使用FileOutputStream和OutputStreamReader类
读文件	使用FileInputStream和InputStreamReader类
保存到外部存储器	使用getExternalStorageDirectory()方法返回指向外部存储器的路径
访问位于res/raw文件夹下的文件	使用Resources对象(通过getResources()方法获得)的openRawResource()方法
创建数据库辅助类	扩展SQLiteOpenHelper类

第7章

内容提供者

本章将介绍以下内容

- 内容提供者简介
- 如何在Android中使用内容提供者
- 如何创建自己的内容提供者
- 如何使用自己的内容提供者

在第6章中，我们学习了持久化数据的不同方法——使用共享首选项、文件系统以及SQLite数据库。虽然使用数据库方法来保存结构化的复杂数据是值得推荐的方式，但数据共享是一个挑战，因为数据库只能被创建它的包访问。

我们将在本章学习Android通过使用内容提供者来共享数据的方法。您将学习到如何使用内置的内容提供者以及通过实现自己的内容提供者来跨包共享数据。

7.1　在Android中共享数据

在Android中，推荐使用内容提供者来实现跨包的数据共享。可以将内容提供者视为一种数据存储。它存储数据的方式和使用它的应用程序无关，重要的是包如何以一致的编程接口来访问存储其中的数据。内容提供者的行为方式与数据库很像——您可以查询它，编辑以及增加或删除其内容。然而，与数据库不同的是，内容提供者可以使用不同的方式来存储数据。数据可以存储于数据库、文件中甚至网络上。

Android附带了许多有用的内容提供者，包括：

- Browser——存储诸如浏览器书签、浏览器历史记录等数据
- CallLog——存储诸如未接电话、通话详细信息等数据
- Contacts——存储联系人详细信息
- MediaStore——存储媒体文件，如音频、视频和图像
- Settings——存储设备的设置和首选项

除了一些内置的以外，还可以创建自己的内容提供者。

为了查询内容提供者，可为特定行使用一个可选的说明符，以URI形式指定查询字符串。查询URI的格式如下所示：

```
<standard_prefix>://<authority>/<data_path>/<id>
```

URI的不同部分如下所示：

- 内容提供者的standard_prefix始终是content://。
- authority指定了内容提供者的名称，如内置的Contacts内容提供者的名称为contacts。对于第三方内容提供者，将采用完全限定的名称，如com.wrox.provider或者net.learn2develop.provider。
- data_path指定了请求数据的类型。例如，如果您是从Contacts内容提供者获取所有联系人，那么data_path就是people，而URI为content://contacts/people。
- id指定了请求的特定记录。例如，如果您从Contacts内容提供者中查找2号联系人，那么URI为content://contacts/people/2。

表7-1列出了一些查询字符串的示例。

表7-1　查询字符串的示例

查询字符串	描　述
content://media/internal/images	返回一个存储在设备上的所有内部图像的列表
content://media/external/images	返回一个存储在设备的外部存储器(如SD卡)上的所有图像的列表
content://call_log/calls	返回一个在CallLog中记录的所有通话的列表
content://browser/bookmarks	返回一个存储在浏览器中的书签列表

7.2　使用内容提供者

理解内容提供者的最佳方法就是实际地运用它。下面的“试一试”展示了在Android应用程序中如何使用内容提供者。

试一试　使用Contacts内容提供者

Provider.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，按图7-1所示创建并命名一个新的Android项目。

(2) 在main.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
>

    <ListView
        android:id="@+id/android:list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
```

```
            android:stackFromBottom="false"
            android:transcriptMode="normal"
            />
        <TextView
            android:id="@+id/contactName"
            android:textStyle="bold"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            />
        <TextView
            android:id="@+id/contactID"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
           />

</LinearLayout>
```

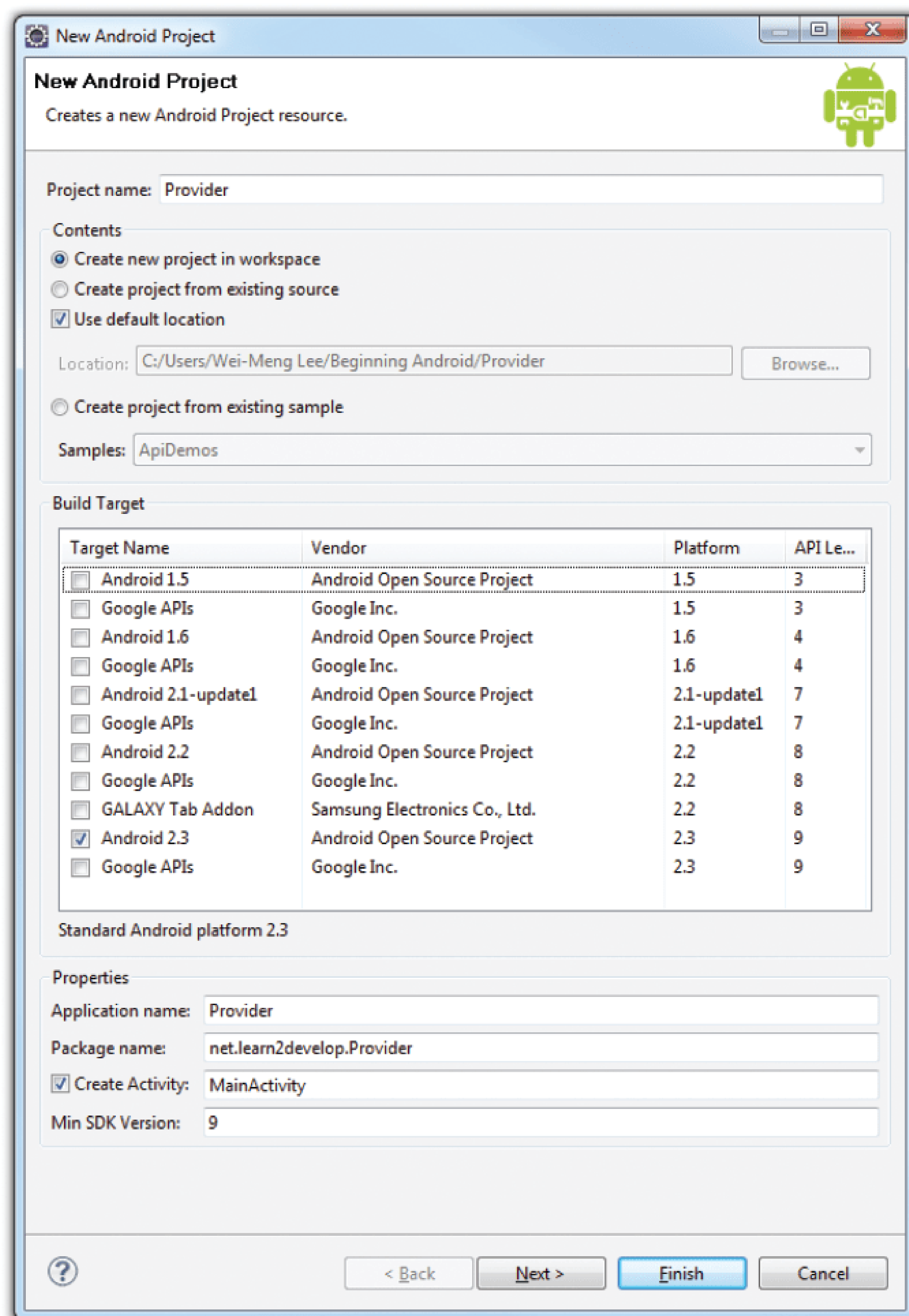


图 7-1

(3) 在MainActivity.java类中编写如下代码：

```
package net.learn2develop.Provider;

import android.app.Activity;
import android.os.Bundle;

import android.app.ListActivity;
import android.database.Cursor;
import android.net.Uri;
import android.provider.ContactsContract;
import android.widget.SimpleCursorAdapter;

public class MainActivity extends ListActivity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Uri allContacts = Uri.parse("content://contacts/people");

        Cursor c = managedQuery(allContacts, null, null, null, null);

        String[] columns = new String[] {
            ContactsContract.Contacts.DISPLAY_NAME,
            ContactsContract.Contacts._ID};
        int[] views = new int[] {R.id.contactName, R.id.contactID};

        SimpleCursorAdapter adapter =
            new SimpleCursorAdapter(this, R.layout.main, c, columns, views);
        this.setListAdapter(adapter);
    }
}
```

(4) 在AndroidManifest.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Provider"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

```
        <uses-sdk android:minSdkVersion="7" />
        <uses-permission android:name="android.permission.READ_CONTACTS">
        </uses-permission>
    </manifest>
```

(5) 启动一个AVD并在Android模拟器中创建一些联系人(如图7-2所示)。

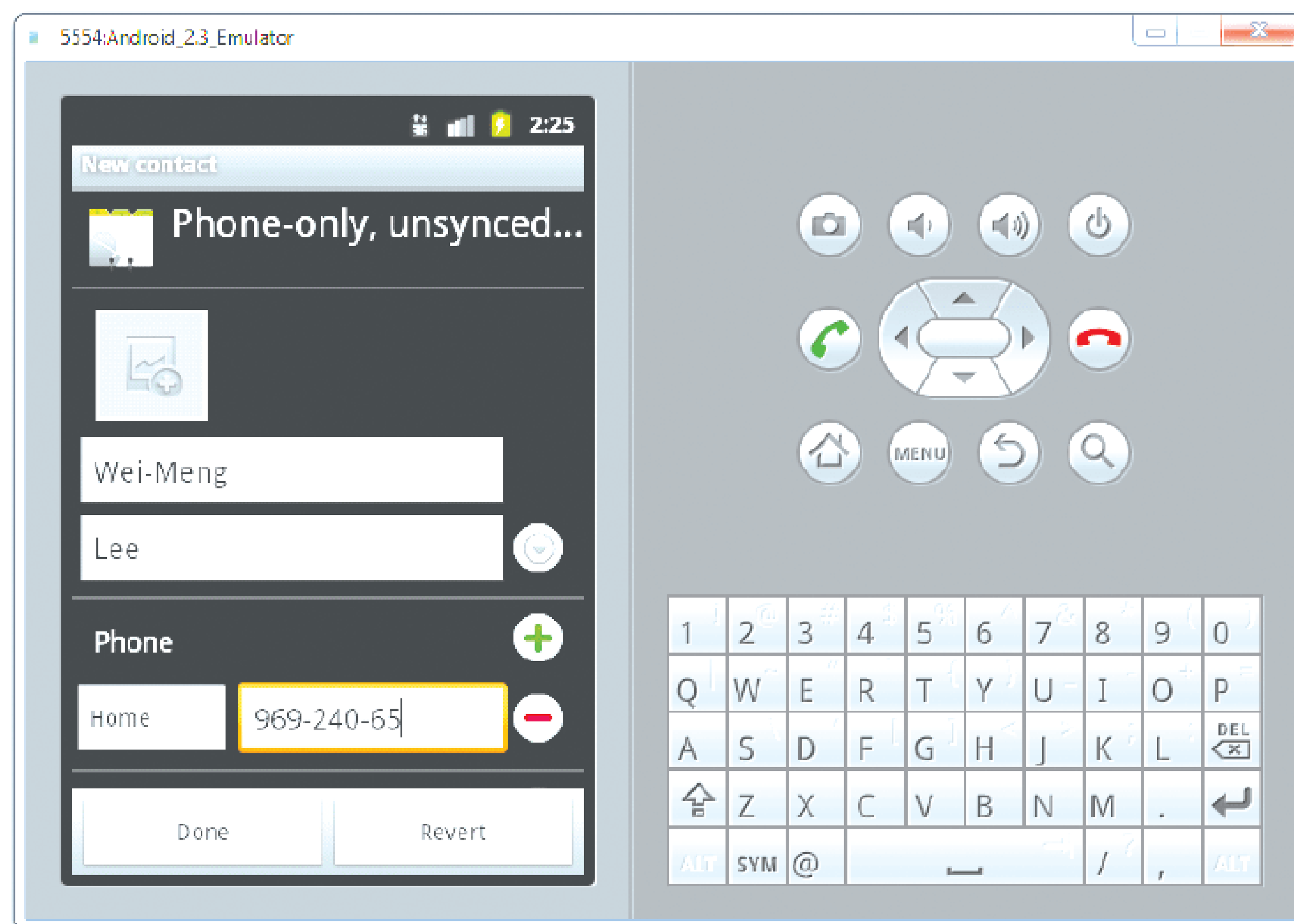


图 7-2

(6) 按F11键在Android模拟器上调试应用程序。图7-3展示了活动显示刚刚创建的联系人列表。

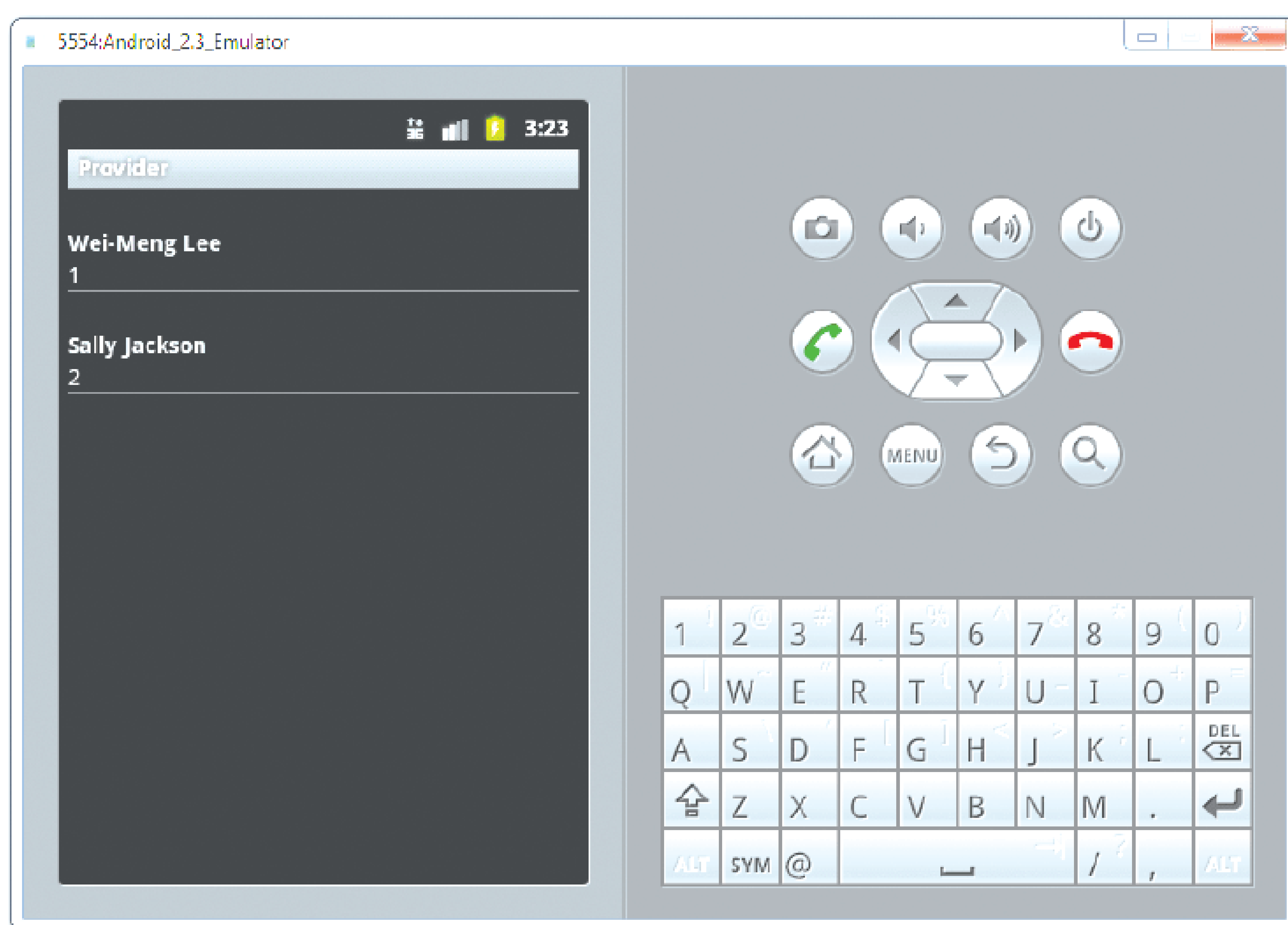


图 7-3

示例说明

在这一示例中，检索所有存储在Contacts应用程序中的联系人并在ListView中进行显示。

Activity类的managedQuery()方法检索一个托管游标(managed cursor)。托管游标处理在应用程序暂停时卸载其自身和在应用程序重启时重新查询自身的所有工作。

语句：

```
Cursor c = managedQuery(allContacts, null, null, null, null);
```

等价于：

```
Cursor c = getContentResolver().query(allContacts, null, null, null, null);
startManagingCursor(c); //---允许活动基于其生命周期来管理游标的生命周期---
```

getContentResolver()方法返回一个ContentResolver对象，它可以使用合适的内容提供者来帮助解析内容URI。

SimpleCursorAdapter对象将一个游标映射到在XML文件(main.xml)中定义的TextView(或者ImageView)，并将数据(由columns表示)映射到视图(由views表示)上：

```
String[] columns = new String[] {
    ContactsContract.Contacts.DISPLAY_NAME,
    ContactsContract.Contacts._ID};
int[] views = new int[] {R.id.contactName, R.id.contactID};

SimpleCursorAdapter adapter =
    new SimpleCursorAdapter(this, R.layout.main, c, columns, views);
this.setListAdapter(adapter);
```

注意，为了使应用程序可以访问Contacts程序，需要在AndroidManifest.xml文件中有READ_CONTACTS权限。

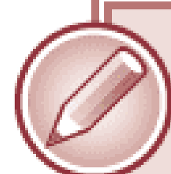
7.2.1　预定义查询字符串常量

除了使用查询URI，还可以利用Android中的一个预定义查询字符串常量的列表来为不同数据类型指定URI。例如，除了使用查询content://contacts/people，还可以使用Android中的一个预定义常量将以下语句：

```
Uri allContacts = Uri.parse("content://contacts/people");
```

重写为：

```
Uri allContacts = ContactsContract.Contacts.CONTENT_URI;
```



注意：对于Android 2.0或更高版本，需要使用ContactsContract.Contacts.CONTENT_URI这个URI来查询基本的Contacts记录。

以下是一些预定义查询字符串常量的示例：

- Browser.BOOKMARKS_URI
- Browser.SEARCHES_URI

- CallLog.CONTENT_URI
- MediaStore.Images.Media.INTERNAL_CONTENT_URI
- MediaStore.Images.Media.EXTERNAL_CONTENT_URI
- Settings.CONTENT_URI

如果想要检索第一个联系人，则可按如下方式指定此联系人的ID：

```
Uri oneContact = Uri.parse("content://contacts/people/1");
```

或者，可以结合ContentUris类的withAppendedId()方法来使用预定义常量：

```
import android.content.ContentUris;
//...
Uri oneContact = ContentUris.withAppendedId(
    ContactsContract.Contacts.CONTENT_URI, 1);
```

除了绑定到ListView，还可以使用Cursor对象将结果输出来，如下所示：

```
package net.learn2develop.Provider;

import android.app.Activity;
import android.os.Bundle;

import android.app.ListActivity;
import android.database.Cursor;
import android.net.Uri;
import android.provider.ContactsContract;
import android.widget.SimpleCursorAdapter;

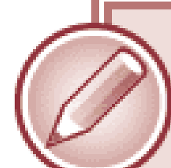
import android.util.Log;

public class MainActivity extends ListActivity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Uri allContacts = ContactsContract.Contacts.CONTENT_URI;
        Cursor c = managedQuery(
            allContacts, null, null, null, null);
        String[] columns = new String[] {
            ContactsContract.Contacts.DISPLAY_NAME,
            ContactsContract.Contacts._ID};
        int[] views = new int[] {R.id.contactName, R.id.contactID};

        SimpleCursorAdapter adapter =
            new SimpleCursorAdapter(this, R.layout.main,
            c, columns, views);
        this.setListAdapter(adapter);
        PrintContacts(c);
    }
```

```
        private void PrintContacts(Cursor c)
        {
            if (c.moveToFirst()) {
                do{
                String contactID = c.getString(c.getColumnIndex(
                        ContactsContract.Contacts._ID));
                String contactDisplayName =
                      c.getString(c.getColumnIndex(
                          ContactsContract.Contacts.DISPLAY_NAME));
                    Log.v("Content Providers", contactID + ", " +
                        contactDisplayName);
                } while (c.moveToNext());
            }
        }
    }
```



注意： 如果对如何查看LogCat窗口不太熟悉，可参阅附录A快速了解一下Eclipse IDE。

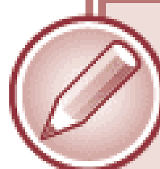
PrintContacts()方法将在LogCat窗口中输出如下内容：

```
12-13 02:40:36.825: VERBOSE/Content Providers(497):
    1, Wei-Meng Lee
12-13 02:40:36.825: VERBOSE/Content Providers(497):
     2, Sally Jackson
```

它输出了存储在Contacts应用程序中的每一个联系人的ID和姓名。这时，通过访问ContactsContract.Contacts._ID字段来获取联系人的ID，访问ContactsContract.Contacts.DISPLAY_NAME来得到联系人的姓名。如果想显示联系人的电话号码，由于这个信息存储于另外一张表中，因此需要再次查询内容提供者：

```
        private void PrintContacts(Cursor c)
        {
            if (c.moveToFirst()) {
                do{
                    String contactID = c.getString(c.getColumnIndex(
                        ContactsContract.Contacts._ID));
                    String contactDisplayName =
                        c.getString(c.getColumnIndex(
                            ContactsContract.Contacts.DISPLAY_NAME));
                    Log.v("Content Providers", contactID + ", " +
                        contactDisplayName);
                    //---获取电话号码---
                    int hasPhone =
                        c.getInt(c.getColumnIndex(
                            ContactsContract.Contacts.HAS_PHONE_NUMBER));
                    if (hasPhone == 1) {
                        Cursor phoneCursor =
                            getContentResolver().query(
```

```
                    ContactsContract.CommonDataKinds.Phone.CONTENT_
                    URI, null,
                    ContactsContract.CommonDataKinds.Phone.CONTACT_
                    ID + " = " + contactID, null, null);
                while (phoneCursor.moveToNext()) {
                    Log.v("Content Providers",
                      phoneCursor.getString(
                          phoneCursor.getColumnIndex(
                          ContactsContract.CommonDataKinds.Phone.
                          NUMBER)));
                }
                phoneCursor.close();
            }
        } while (c.moveToNext());
    }
  }
}
```



注意： 为了访问一个联系人的电话号码，需要对存储于ContactsContract.CommonDataKinds.Phone.CONTENT_URI中的URI进行查询。

在上面的代码片段中，首先使用ContactsContract.Contacts.HAS_PHONE_NUMBER字段检查一个联系人是否有电话号码。如果其至少有一个电话号码，就可以基于此联系人的ID对内容提供者再次查询。一旦检索到这一(些)号码，就可以迭代遍历并把它们输出来。应有如下显示内容：

```
12-13 02:41:09.541: VERBOSE/Content Providers(546):
    1, Wei-Meng Lee
12-13 02:41:09.541: VERBOSE/Content Providers(546):
    969-240-65
12-13 02:41:09.541: VERBOSE/Content Providers(546):
    2, Sally Jackson
12-13 02:41:09.541: VERBOSE/Content Providers(546):
    345-668-43
```

7.2.2 投影

managedQuery()方法的第2个参数控制查询返回的列数。这个参数被称为投影(projection)，之前是指定为null：

```
Cursor c = managedQuery(allContacts,
    null, null, null, null);
```

可以通过创建一个包含需要返回的列名的数组来指定要返回的确切列，如下所示：

```
String[] projection = new String[]
    {ContactsContract.Contacts._ID,
     ContactsContract.Contacts.DISPLAY_NAME,
     ContactsContract.Contacts.HAS_PHONE_NUMBER};
Cursor c = managedQuery(allContacts, projection,
    null, null, null);
```

在上述情况下，_ID、DISPLAY_NAME以及HAS_PHONE_NUMBER字段都将被检索。

7.2.3　筛选

managedQuery()方法的第3个和第4个参数可以用来指定一个SQL的WHERE子句来对查询结果进行筛选。例如，下列语句只检索名字以Lee结尾的人：

```
Cursor c = managedQuery(allContacts, projection,
    ContactsContract.Contacts.DISPLAY_NAME + " LIKE '%Lee'",
    null, null);
```

这里，第3个参数包含了一个SQL语句，其中含有要搜索的名字(Lee)。还可以将搜索字符串放在方法的第4个参数中，如下所示：

```
Cursor c = managedQuery(allContacts, projection,
    ContactsContract.Contacts.DISPLAY_NAME + " LIKE ?",
    new String[] {"%Lee"} , null);
```

7.2.4　排序

managedQuery()方法的第5个参数可以用来指定一个SQL的ORDER BY子句来对查询结果排序。例如，下列语句将联系人名字按升序进行排序：

```
Cursor c = managedQuery(
    allContacts,
    projection,
    ContactsContract.Contacts.DISPLAY_NAME + " LIKE ?",
    new String[] {"%"} ,
    ContactsContract.Contacts.DISPLAY_NAME + " ASC");
```

若要按降序排列，则使用DESC关键字：

```
Cursor c = managedQuery(
    allContacts,
    projection,
    ContactsContract.Contacts.DISPLAY_NAME + " LIKE ?",
    new String[] {"%"} ,
    ContactsContract.Contacts.DISPLAY_NAME + " DESC");
```

7.3　创建自己的内容提供者

在Android中创建自己的内容提供者非常简单。所有您需要做的是扩展抽象类ContentProvider，并重写其中定义的各种方法。

本节将学习如何创建一个简单的内容提供者来存储一个图书列表。为了便于说明，内容提供者将图书存储在一个包含3个字段的数据库表中，如图7-4所示。

_id	title	isbn

图 7-4

下面的“试一试”展示了具体的操作步骤。

试一试　创建自己的内容提供者

ContentProviders.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个新的Android项目并命名为ContentProviders。

(2) 在项目的src文件夹下增加一个新的Java类文件，并命名为BooksProvider.java(如图7-5所示)。

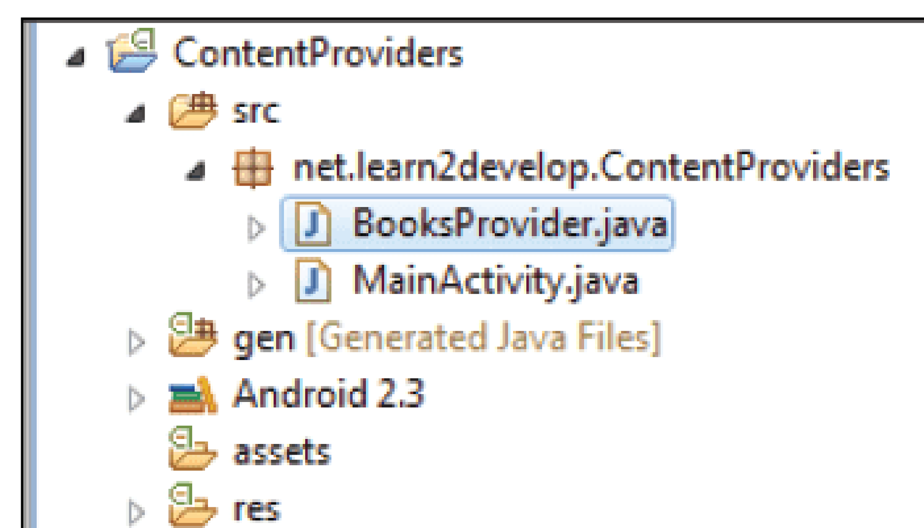


图 7-5

(3) 按如下所示填充BooksProvider.java文件：

```
package net.learn2develop.ContentProviders;

import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import android.text.TextUtils;
import android.util.Log;

public class BooksProvider extends ContentProvider
{
    public static final String PROVIDER_NAME =
        "net.learn2develop.provider.Books";

    public static final Uri CONTENT_URI =
        Uri.parse("content://"+ PROVIDER_NAME + "/books");

    public static final String _ID = "_id";
    public static final String TITLE = "title";
    public static final String ISBN = "isbn";

    private static final int BOOKS = 1;
    private static final int BOOK_ID = 2;

    private static final UriMatcher uriMatcher;
    static{
        uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        uriMatcher.addURI(PROVIDER_NAME, "books", BOOKS);
        uriMatcher.addURI(PROVIDER_NAME, "books/#", BOOK_ID);
    }

    //---为了使用数据库---
```

```
private SQLiteDatabase booksDB;
private static final String DATABASE_NAME = "Books";
private static final String DATABASE_TABLE = "titles";
private static final int DATABASE_VERSION = 1;
private static final String DATABASE_CREATE =
        "create table " + DATABASE_TABLE +
        " (_id integer primary key autoincrement, "
        + "title text not null, isbn text not null);";

private static class DatabaseHelper extends SQLiteOpenHelper
{
    DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db)
    {
        db.execSQL(DATABASE_CREATE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion,
    int newVersion) {
        Log.w("Content provider database",
              "Upgrading database from version " +
              oldVersion + " to " + newVersion +
              ", which will destroy all old data");
        db.execSQL("DROP TABLE IF EXISTS titles");
        onCreate(db);
    }
}
@Override
public int delete(Uri arg0, String arg1, String[] arg2) {
    // arg0 = uri
    // arg1 = selection
    // arg2 = selectionArgs
    int count=0;
    switch (uriMatcher.match(arg0)){
        case BOOKS:
            count = booksDB.delete(
                    DATABASE_TABLE,
                    arg1,
                    arg2);
            break;
        case BOOK_ID:
            String id = arg0.getPathSegments().get(1);
            count = booksDB.delete(
                    DATABASE_TABLE,
                    _ID + " = " + id +
```

```
                    (!TextUtils.isEmpty(arg1) ? " AND (" +
                            arg1 + ')' : ""),
                            arg2);
                break;
            default: throw new IllegalArgumentException("Unknown URI " + arg0);
        }
        getContext().getContentResolver().notifyChange(arg0, null);
        return count;
    }

    @Override
    public String getType(Uri uri) {
        switch (uriMatcher.match(uri)){
            //---获取所有图书---
            case BOOKS:
                return "vnd.android.cursor.dir/vnd.learn2develop.books ";
            //---获取特定的一本图书---
            case BOOK_ID:
                return "vnd.android.cursor.item/vnd.learn2develop.books ";
            default:
                throw new IllegalArgumentException("Unsupported URI: " + uri);
        }
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        //---添加一本新图书---
        long rowID = booksDB.insert(
            DATABASE_TABLE,
            "",
            values);

        //---如果成功添加---
        if (rowID>0)
        {
            Uri _uri = ContentUris.withAppendedId(CONTENT_URI, rowID);
            getContext().getContentResolver().notifyChange(_uri, null);
            return _uri;
        }
        throw new SQLException("Failed to insert row into " + uri);
    }

    @Override
    public boolean onCreate() {
        Context context = getContext();
        DatabaseHelper dbHelper = new DatabaseHelper(context);
        booksDB = dbHelper.getWritableDatabase();
        return (booksDB == null)? false:true;
    }
```

```
@Override
public Cursor query(Uri uri, String[] projection, String selection,
        String[] selectionArgs, String sortOrder) {
    SQLiteQueryBuilder sqlBuilder = new SQLiteQueryBuilder();
    sqlBuilder.setTables(DATABASE_TABLE);

    if (uriMatcher.match(uri) == BOOK_ID)
        //---如果是获取特定的一本图书---
        sqlBuilder.appendWhere(
            _ID + " = " + uri.getPathSegments().get(1));

    if (sortOrder==null || sortOrder=="")
        sortOrder = TITLE;

    Cursor c = sqlBuilder.query(
            booksDB,
            projection,
            selection,
            selectionArgs,
            null,
            null,
            sortOrder);

    //---注册以便观察内容URI的变化---
    c.setNotificationUri(getContext().getContentResolver(), uri);
    return c;
}

@Override
public int update(Uri uri, ContentValues values, String selection,
        String[] selectionArgs) {
    int count = 0;
    switch (uriMatcher.match(uri)){
        case BOOKS:
            count = booksDB.update(
                    DATABASE_TABLE,
                    values,
                    selection,
                    selectionArgs);
            break;
        case BOOK_ID:
            count = booksDB.update(
                    DATABASE_TABLE,
                    values,
                    _ID + " = " + uri.getPathSegments().get(1) +
                    (!TextUtils.isEmpty(selection) ? " AND (" +
                        selection + ')' : ""),
                    selectionArgs);
            break;
        default: throw new IllegalArgumentException("Unknown URI " + uri);
```

```
            }
            getContext().getContentResolver().notifyChange(uri, null);
            return count;
        }
    }
```

(4) 在AndroidManifest.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.ContentProviders"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
                />
            </intent-filter>
        </activity>
        <provider android:name="BooksProvider"
                android:authorities="net.learn2develop.provider.Books" />
    </application>
    <uses-sdk android:minSdkVersion="9" />
</manifest>
```

示例说明

在本例中，首先创建一个名为BooksProvider的类，它扩展了ContentProvider基类。这个类中重写的各个方法如下所示：

- getType()——返回给定URI上的数据的MIME类型
- onCreate()——当启动提供者时调用
- query()——接收客户端请求，结果以Cursor对象形式返回
- insert()——向内容提供者中插入一条新记录
- delete()——从内容提供者中删除一条现有记录
- update()——在内容提供者中更新一条现有记录

在内容提供者中，可以自由选择如何存储数据——传统的文件系统、XML、数据库或者通过Web服务。本例中，使用前一章讨论的SQLite数据库方法。

接着在BooksProvider类中定义以下常量：

```
public static final String PROVIDER_NAME =
    "net.learn2develop.provider.Books";

public static final Uri CONTENT_URI =
    Uri.parse("content://"+ PROVIDER_NAME + "/books");
```

```
public static final String _ID = "_id";
public static final String TITLE = "title";
public static final String ISBN = "isbn";

private static final int BOOKS = 1;
private static final int BOOK_ID = 2;

private static final UriMatcher uriMatcher;
static{
    uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
    uriMatcher.addURI(PROVIDER_NAME, "books", BOOKS);
    uriMatcher.addURI(PROVIDER_NAME, "books/#", BOOK_ID);
}
```

在上面的代码中可看到，使用一个UriMatcher对象来解析通过一个ContentResolver传递给内容提供者的内容URI。例如，下面的内容URI代表了一个对内容提供者中的所有图书的请求：

```
content://net.learn2develop.provider.Books/books
```

下面的内容URI代表了一个对_id为5的特定图书的请求：

```
content://net.learn2develop.provider.MailingList/books/5
```

内容提供者使用SQLite数据库存储图书。注意，使用SQLiteOpenHelper辅助类来协助管理数据库：

```
private static class DatabaseHelper extends SQLiteOpenHelper
{
    DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db)
    {
        db.execSQL(DATABASE_CREATE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion,
    int newVersion) {
        Log.w("Content provider database",
              "Upgrading database from version " +
              oldVersion + " to " + newVersion +
              ", which will destroy all old data");
        db.execSQL("DROP TABLE IF EXISTS titles");
        onCreate(db);
    }
}
```

接下来，通过重写getType()方法来唯一地描述内容提供者的数据类型。使用UriMatcher对

象，对于单本图书返回vnd.android.cursor.item/vnd.learn2develop.books，对于多本图书返回vnd.android.cursor.dir/vnd.learn2develop.books：

```
@Override
public String getType(Uri uri) {
     switch (uriMatcher.match(uri)){
         //---获取所有图书---
         case BOOKS:
             return "vnd.android.cursor.dir/vnd.learn2develop.books ";
         //---获取特定的一本图书---
         case BOOK_ID:
             return "vnd.android.cursor.item/vnd.learn2develop.books ";
         default:
             throw new IllegalArgumentException("Unsupported URI: " + uri);
     }
 }
```

下一步，重写onCreate()方法以在内容提供者启动时打开一个到数据库的连接：

```
@Override
public boolean onCreate() {
    Context context = getContext();
    DatabaseHelper dbHelper = new DatabaseHelper(context);
    booksDB = dbHelper.getWritableDatabase();
    return (booksDB == null)? false:true;
}
```

重写query()方法，以允许客户端查询图书：

```
@Override
public Cursor query(Uri uri, String[] projection, String selection,
        String[] selectionArgs, String sortOrder) {
    SQLiteQueryBuilder sqlBuilder = new SQLiteQueryBuilder();
    sqlBuilder.setTables(DATABASE_TABLE);

    if (uriMatcher.match(uri) == BOOK_ID)
        //---如果是获取特定的一本图书---
        sqlBuilder.appendWhere(
            _ID + " = " + uri.getPathSegments().get(1));

    if (sortOrder==null || sortOrder=="")
        sortOrder = TITLE;

    Cursor c = sqlBuilder.query(
            booksDB,
            projection,
            selection,
            selectionArgs,
            null,
            null,
            sortOrder);
```

```
        //---注册以便观察内容URI的变化---
        c.setNotificationUri(getContext().getContentResolver(), uri);
        return c;
    }
```

默认情况下，查询的结果按title字段进行排序。查询结果以Cursor对象形式返回。

为了在内容提供者中插入一本新图书，需要重写insert()方法：

```
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        //---添加一本新图书---
        long rowID = booksDB.insert(
            DATABASE_TABLE,
            "",
            values);

        //---如果成功添加---
        if (rowID>0)
        {
            Uri _uri = ContentUris.withAppendedId(CONTENT_URI, rowID);
            getContext().getContentResolver().notifyChange(_uri, null);
            return _uri;
        }
        throw new SQLException("Failed to insert row into " + uri);
    }
```

一旦记录被成功插入，则调用ContentResolver的notifyChange()方法。这将通知已注册的观察者更新了一行。

若要删除一本图书，则重写delete()方法如下：

```
    public int delete(Uri arg0, String arg1, String[] arg2) {
        // arg0 = uri
        // arg1 = selection
        // arg2 = selectionArgs
        int count=0;
        switch (uriMatcher.match(arg0)){
            case BOOKS:
                count = booksDB.delete(
                        DATABASE_TABLE,
                        arg1,
                        arg2);
                break;
            case BOOK_ID:
                String id = arg0.getPathSegments().get(1);
                count = booksDB.delete(
                        DATABASE_TABLE,
                        _ID + " = " + id +
                        (!TextUtils.isEmpty(arg1) ? " AND (" +
                                arg1 + ')' : ""),
                                arg2);
```

```
                break;
            default: throw new IllegalArgumentException("Unknown URI " + arg0);
        }
        getContext().getContentResolver().notifyChange(arg0, null);
        return count;
    }
```

同样，在删除操作后要调用ContentResolver的notifyChange()方法。这将通知已注册的观察者删除了一行。

最后，若要更新一本图书，则重写update()方法如下：

```
    @Override
    public int update(Uri uri, ContentValues values, String selection,
            String[] selectionArgs) {
        int count = 0;
        switch (uriMatcher.match(uri)){
            case BOOKS:
                count = booksDB.update(
                        DATABASE_TABLE,
                        values,
                        selection,
                        selectionArgs);
                break;
            case BOOK_ID:
                count = booksDB.update(
                        DATABASE_TABLE,
                        values,
                        _ID + " = " + uri.getPathSegments().get(1) +
                        (!TextUtils.isEmpty(selection) ? " AND (" +
                            selection + ')' : ""),
                        selectionArgs);
                break;
            default: throw new IllegalArgumentException("Unknown URI " + uri);
        }
        getContext().getContentResolver().notifyChange(uri, null);
        return count;
    }
}
```

与insert()和delete()方法一样，在更新后调用ContentResolver的notifyChange()方法。这将通知已注册的观察者更新了一行。

最后，为了将内容提供者注册到Android，可以修改AndroidManifest.xml文件，添加<provider>元素。

使用内容提供者

既然已经构建了自己的新的内容提供者，那就可以在Android应用程序中测试它。下面的“试一试”展示了是如何做到这一点的。

试一试　使用新建的内容提供者

(1) 使用在前一节中所创建的同一个项目，在main.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="ISBN" />

<EditText
    android:id="@+id/txtISBN"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent" />

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Title" />

<EditText
    android:id="@+id/txtTitle"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent" />

<Button
    android:text="Add title"
    android:id="@+id/btnAdd"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

<Button
    android:text="Retrieve titles"
    android:id="@+id/btnRetrieve"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

</LinearLayout>
```

(2) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.ContentProviders;

import android.app.Activity;
```

```
import android.os.Bundle;

import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import android.content.ContentValues;
import android.database.Cursor;
import android.net.Uri;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button btnAdd = (Button) findViewById(R.id.btnAdd);
        btnAdd.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //---添加一本图书---
                ContentValues values = new ContentValues();
                values.put(BooksProvider.TITLE, ((EditText)
                    findViewById(R.id.txtTitle)).getText().toString());
                values.put(BooksProvider.ISBN, ((EditText)
                    findViewById(R.id.txtISBN)).getText().toString());
                Uri uri = getContentResolver().insert(
                    BooksProvider.CONTENT_URI, values);
                Toast.makeText(getBaseContext(),uri.toString(),
                    Toast.LENGTH_LONG).show();
            }
        });

        Button btnRetrieve = (Button) findViewById(R.id.btnRetrieve);
        btnRetrieve.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //---检索标题---
                Uri allTitles = Uri.parse(
                    "content://net.learn2develop.provider.Books/books");
                Cursor c = managedQuery(allTitles, null, null, null,
                    "title desc");
                if (c.moveToFirst()) {
                    do{
                        Log.v("ContentProviders",
                            c.getString(c.getColumnIndex(
                                BooksProvider._ID)) + ", " +
                            c.getString(c.getColumnIndex(
                                BooksProvider.TITLE)) + ", " +
                            c.getString(c.getColumnIndex(
```

```
                                    BooksProvider.ISBN)));
                        } while (c.moveToNext());
                    }
                }
            });
        }
    }
```

(3) 按F11键在Android模拟器上调试应用程序。

(4) 输入一本图书的ISBN和标题并单击Add按钮。图7-6展示了由Toast类显示添加到内容提供者的图书的URI。为了检索存储在内容提供者中的所有标题，单击Retrieve titles按钮并观察在Eclipse的LogCat窗口中所输出的值。

示例说明

首先，修改活动，这样用户就可以输入一本图书的ISBN和标题来添加到刚刚创建的内容提供者中。

要添加一本图书到内容提供者中，可以创建一个新的ContentValues对象，然后用与此图书有关的各种信息来填充这一对象：

```
//---添加一本图书---
ContentValues values = new ContentValues();
values.put(BooksProvider.TITLE, ((EditText)
    findViewById(R.id.txtTitle)).getText().toString());
values.put(BooksProvider.ISBN, ((EditText)
    findViewById(R.id.txtISBN)).getText().toString());
Uri uri = getContentResolver().insert(
    BooksProvider.CONTENT_URI, values);
```

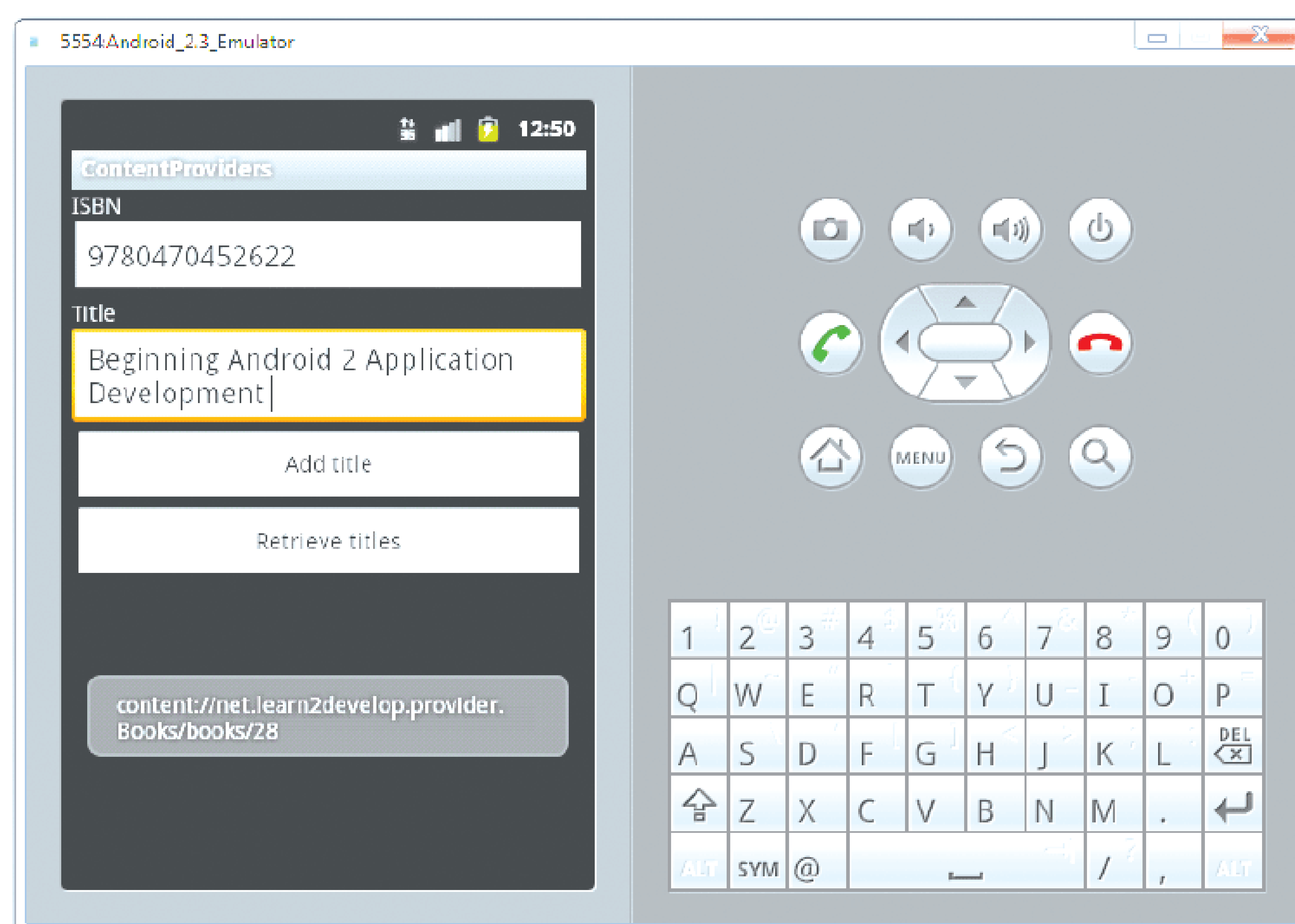


图 7-6

注意，因为内容提供者是在同一个包中，所以可以分别使用BooksProvider.TITLE和BooksProvider.ISBN常量来表示title和isbn字段。如果是从另一个包来访问内容提供者的，那么将不能够使用这些常量。在那种情况下，需要直接指定字段名称，如下所示：

```
ContentValues values = new ContentValues();
values.put("title", ((EditText)
    findViewById(R.id.txtTitle)).getText().toString());
values.put("isbn", ((EditText)
    findViewById(R.id.txtISBN)).getText().toString());
Uri uri = getContentResolver().insert(
        Uri.parse(
        "content://net.learn2develop.provider.Books/books"),
        values);
Toast.makeText(getBaseContext(),uri.toString(),
    Toast.LENGTH_LONG).show();
```

另外还要注意，对于外部包，需要使用完全限定的名称来引用内容URI：

```
Uri.parse(
    "content://net.learn2develop.provider.Books/books"),
```

要检索内容提供者中的所有标题，可使用下面的代码片段：

```
Uri allTitles = Uri.parse(
    "content://net.learn2develop.provider.Books/books");

Cursor c = managedQuery(allTitles, null, null, null,
    "title desc");
if (c.moveToFirst()) {
    do{
        Log.v("ContentProviders",
            c.getString(c.getColumnIndex(
                BooksProvider._ID)) + ", " +
            c.getString(c.getColumnIndex(
                BooksProvider.TITLE)) + ", " +
            c.getString(c.getColumnIndex(
                BooksProvider.ISBN)));
    } while (c.moveToNext());
}
```

前面的查询将返回按title字段降序排列的结果。

如果想更新一本图书的详细信息，调用update()方法，使用内容URI来指示图书的ID，

```
ContentValues editedValues = new ContentValues();
editedValues.put(BooksProvider.TITLE, "Android Tips and Tricks");
getContentResolver().update(
    Uri.parse(
        "content://net.learn2develop.provider.Books/books/2"),
    editedValues,
    null,
    null);
```

要删除一本图书，调用delete()方法，使用内容URI来指示图书的ID：

```
getContentResolver().delete(
        Uri.parse("content://net.learn2develop.provider.Books/books/2"),
        null, null);
```

要删除所有图书，只要在内容URI中省略图书的ID：

```
getContentResolver().delete(
        Uri.parse("content://net.learn2develop.provider.Books/books"),
        null, null);
```

7.4　本章小结

在这一章中，我们了解了什么是内容提供者，以及如何使用一些内置在Android中的内容提供者。特别是，我们学习了如何使用Contacts内容提供者。Google做出的提供内容提供者的决定使得应用程序可以通过一套标准的编程接口进行数据共享。除了内置的内容提供者以外，还可以由自己创建自定义的内容提供者来与其他包实现数据共享。

练　习

1. 写一个查询，实现从Contacts应用程序中检索所有包含单词jack的联系人。
2. 说出在实现自己的内容提供者时必须重写的方法。
3. 如何在AndroidManifest.xml文件中注册一个内容提供者?

练习答案参见附录C。

本章主要内容

主　题	关键概念
检索一个托管游标	使用managedQuery()方法
为内容提供者指定查询的两种方法	使用查询URI或者预定义查询字符串常量
在一个内容提供者中检索一列的值	使用getColumnIndex()方法
访问联系人姓名所用的查询URI	ContactsContract.Contacts.CONTENT_URI
访问联系人电话号码所用的查询URI	ContactsContract.CommonDataKinds.Phone.CONTENT_URI
创建自己的内容提供者	创建一个扩展ContentProvider类的类

第8章 消息传递和联网

本章将介绍以下内容

- 如何以编程方式通过应用程序发送SMS消息
- 如何使用内置的Messaging应用程序发送SMS消息
- 如何接收传入的SMS消息
- 如何通过应用程序发送电子邮件消息
- 如何使用HTTP连接到Web
- 如何使用Web服务

一旦启动并运行基本的Android应用程序，下一个有趣的事情就是为其添加与外界通信的能力。您可能希望在一件事情发生(如您到达了一个特定的地理位置)时应用程序可以给另一部手机发送SMS消息，或者可能希望访问一个提供特定服务(如汇率、天气等)的Web服务。

在本章中，将学习如何以编程方式从Android应用程序中发送和接收SMS消息。

还将学习如何使用HTTP协议来与Web服务器对话，从而可以下载文本和二进制数据。本章的最后部分将介绍如何解析XML文件来提取一个XML文件的相关部分——这是访问Web服务时很有用的一种技术。

8.1 SMS消息传递

SMS消息传递是当今手机上的一个主要的杀手级应用——对于一些用户来说，这跟手机本身一样必不可少。当前您购买的任何手机都至少应该具有SMS消息传递的功能，几乎所有年龄段的用户都知道如何发送和接收这类消息。Android带有一个内置的SMS应用程序，可以接收和发送SMS消息。不过，在某些情况下，您可能想要将SMS功能集成到您自己的应用程序中。举个例子，您也许打算写一个能够按固定时间间隔自动发送SMS消息的应用程序。例如，您想追踪孩子的位置时这就非常有用——只要给他们一个Android设备，可以每30分钟发出一条包含地理位置信息的SMS消息就行了。这下，您就对他们放了学是否真的去了图书馆了解得一清二楚(当然，这也意味着您不得不为发送这类短信而破点费)。

本节介绍如何在Android应用程序中以编程方式发送和接收SMS消息。对Android开发人员来说，好消息是不需要用一个真正的设备来对SMS消息传递进行测试：免费的Android模拟器提供了这一功能。

8.1.1　以编程方式发送SMS消息

首先，您将学习如何以编程方式通过应用程序发送SMS消息。使用这种方法，应用程序可以自动发送短信给收件人，而无须用户干预。下面的“试一试”将告诉您这是如何做到的。

试一试　发送SMS消息

SMS.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，按图8-1所示创建一个新的Android项目。

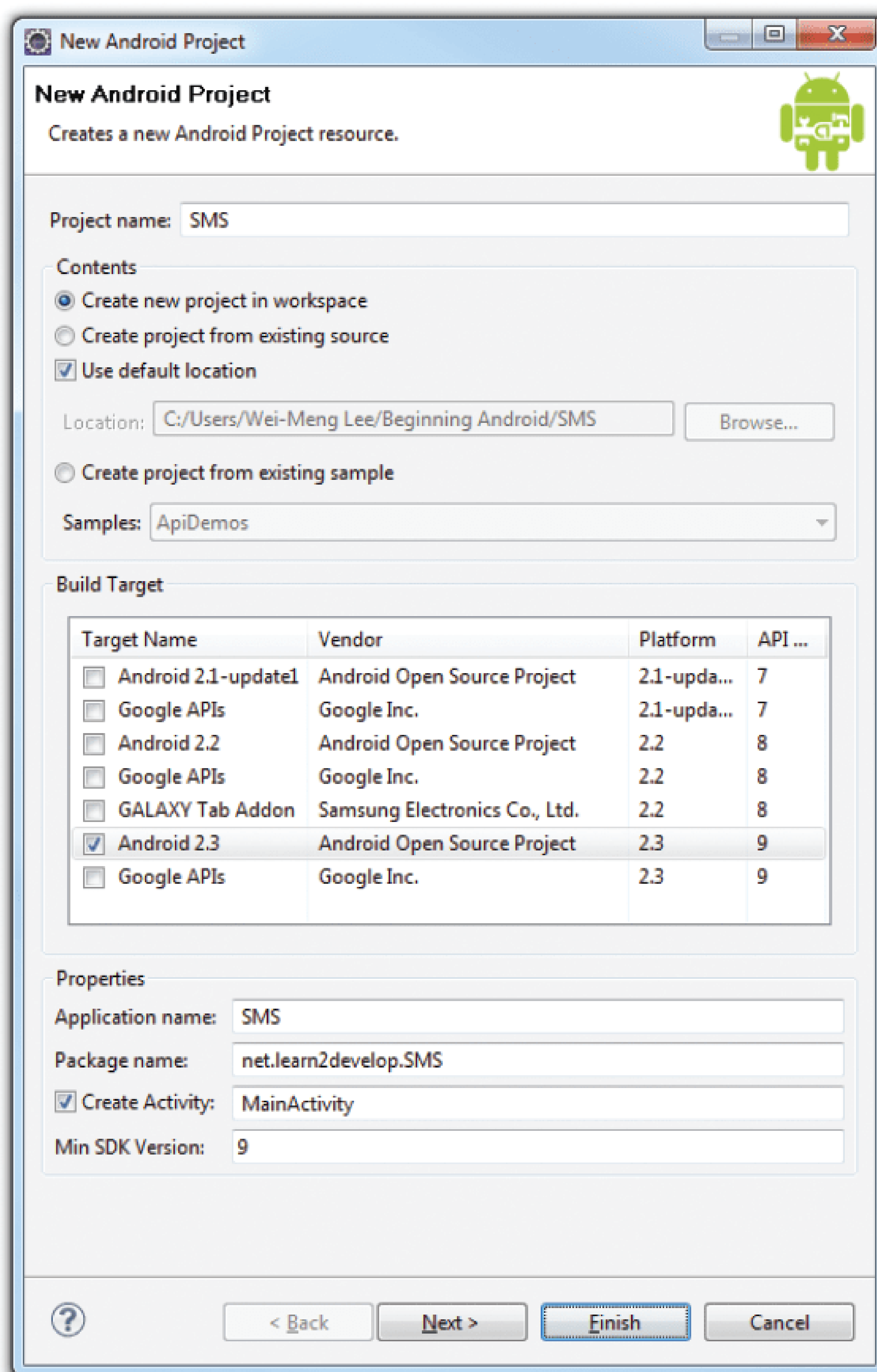


图 8-1

(2) 在main.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
>
```

```
<Button
    android:id="@+id/btnSendSMS"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Send SMS" />

</LinearLayout>
```

(3) 在AndroidManifest.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.SMS"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
</manifest>
```

(4) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.SMS;

import android.app.Activity;
import android.os.Bundle;

import android.app.PendingIntent;
import android.content.Intent;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {
    Button btnSendSMS;
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        btnSendSMS = (Button) findViewById(R.id.btnSendSMS);
```

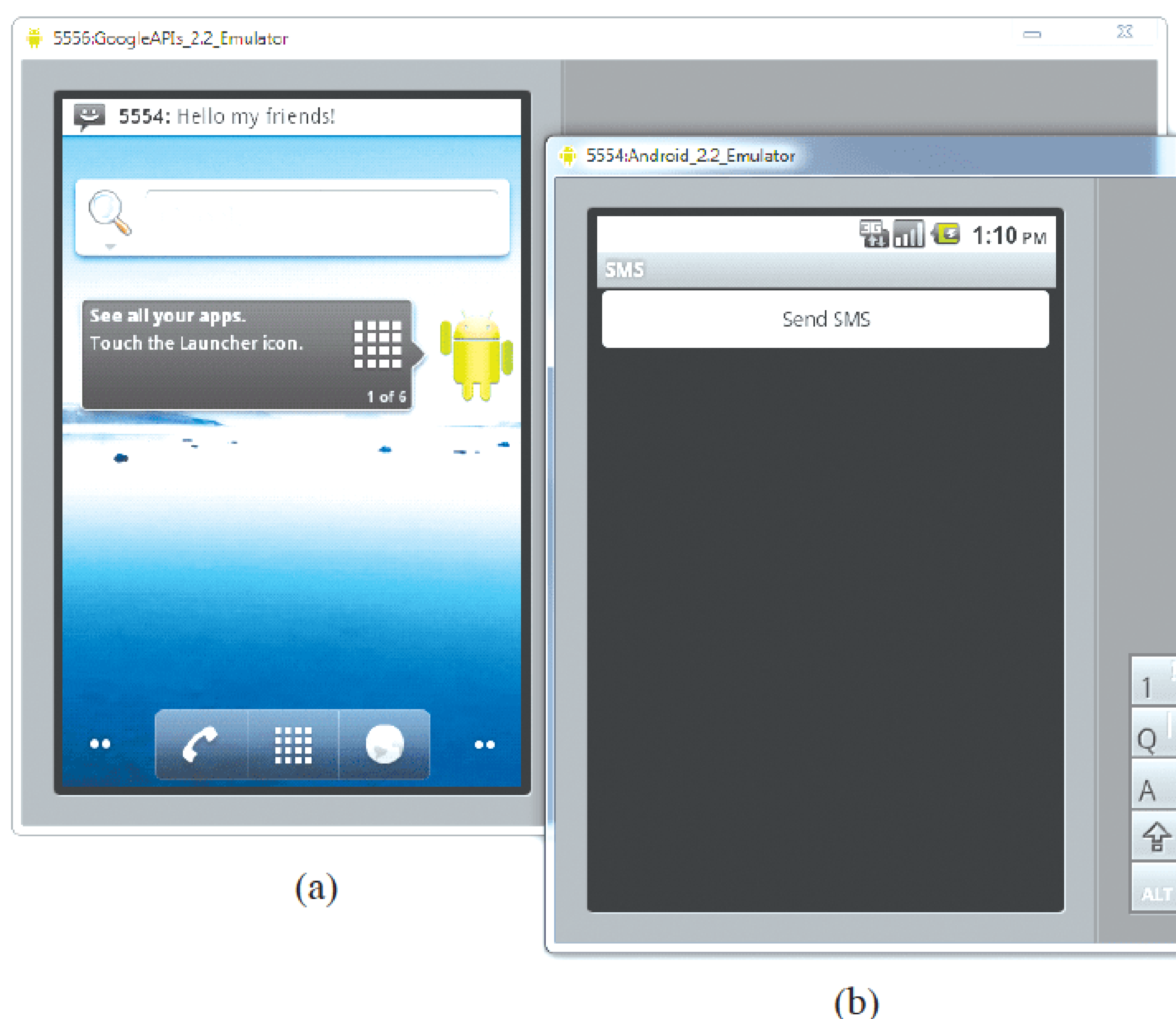
```
            btnSendSMS.setOnClickListener(new View.OnClickListener()
            {
                public void onClick(View v)
                {
                    sendSMS("5556", "Hello my friends!");

                }
            });
        }

        //---发送一条短信到另一个设备---
        private void sendSMS(String phoneNumber, String message)
        {
            SmsManager sms = SmsManager.getDefault();
            sms.sendTextMessage(phoneNumber, null, message, null, null);
        }
    }
```

(5) 按F11键在Android模拟器上调试应用程序。使用Android SDK and AVD Manager启动另一个AVD。

(6) 在第1个Android模拟器上，单击Send SMS按钮来发送SMS消息到第2个模拟器。图8-2(a)展示了由第2个模拟器收到的SMS消息(注意在第2个模拟器顶端的通知栏)。



(a)

(b)

图 8-2

示例说明

Android使用基于权限的策略，因此应用程序所需的所有权限都必须在AndroidManifest.xml文件中指定。这可以确保在应用程序安装时，用户确切地知道它需要哪些访问权限。

由于发送SMS消息会导致用户的额外费用，因此在AndroidManifest.xml文件中指明SMS权

限可以使用户决定是否允许安装应用程序。

使用SmsManager类，可以以编程方式来发送SMS消息。与其他类不同，不能直接实例化这个类，而是要调用getDefault()静态方法获得一个SmsManager对象。然后，使用sendTextMessage()方法来发送SMS消息：

```
private void sendSMS(String phoneNumber, String message)
{
    SmsManager sms = SmsManager.getDefault();
    sms.sendTextMessage(phoneNumber, null, message, null, null);
}
```

以下是sendTextMessage()方法用到的5个参数：

- destinationAddress——收件人的电话号码
- scAddress——服务中心地址，null代表默认的SMSC
- text——SMS消息的内容
- sentIntent——发送消息后调用的挂起的意图(8.1.2节将详细讨论)
- deliveryIntent——消息递送后调用的挂起的意图(8.1.2节详细讨论)

8.1.2 发送消息后获取反馈

8.1.1节学习了如何使用SmsManager类以编程方式发送SMS消息，但如何知道消息已被正确发送了呢？要达到此目的，可以创建两个PendingIntent对象来监视SMS消息发送过程中的状态。这两个PendingIntent对象传递给sendTextMessage()方法的最后两个参数。下面的代码片段展示了如何监视被发送的SMS消息的状态：

```
//---发送一条SMS消息到另一个设备---
private void sendSMS(String phoneNumber, String message)
{
    String SENT = "SMS_SENT";
    String DELIVERED = "SMS_DELIVERED";

    PendingIntent sentPI = PendingIntent.getBroadcast(this, 0,
        new Intent(SENT), 0);

    PendingIntent deliveredPI = PendingIntent.getBroadcast(this, 0,
        new Intent(DELIVERED), 0);

    //---当SMS消息已被发送时---
    registerReceiver(new BroadcastReceiver(){
        @Override
        public void onReceive(Context arg0, Intent arg1) {
            switch (getResultCode())
            {
                case Activity.RESULT_OK:
                    Toast.makeText(getBaseContext(), "SMS sent",
                            Toast.LENGTH_SHORT).show();
                    break;
```

```
                    case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                        Toast.makeText(getBaseContext(), "Generic failure",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_NO_SERVICE:
                        Toast.makeText(getBaseContext(), "No service",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_NULL_PDU:
                        Toast.makeText(getBaseContext(), "Null PDU",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_RADIO_OFF:
                        Toast.makeText(getBaseContext(), "Radio off",
                                Toast.LENGTH_SHORT).show();
                        break;
                }
            }
        }, new IntentFilter(SENT));

        //---当SMS消息已被递送时---
        registerReceiver(new BroadcastReceiver(){
            @Override
            public void onReceive(Context arg0, Intent arg1) {
                switch (getResultCode())
                {
                    case Activity.RESULT_OK:
                        Toast.makeText(getBaseContext(), "SMS delivered",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case Activity.RESULT_CANCELED:
                        Toast.makeText(getBaseContext(), "SMS not delivered",
                                Toast.LENGTH_SHORT).show();
                        break;
                }
            }
        }, new IntentFilter(DELIVERED));

        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, message, sentPI, deliveredPI);
    }
```

在这里，创建了两个PendingIntent对象。然后，为两个BroadcastReceiver进行注册。这两个BroadcastReceiver侦听与SMS_SENT和SMS_DELIVERED匹配的意图(分别在发送和接收消息后由操作系统触发)。在每一个BroadcastReceiver中，重写onReceive()方法并得到当前的结果码。

两个PendingIntent对象被传递给sendTextMessage()方法的最后两个参数：

```
sms.sendTextMessage(phoneNumber, null, message, sentPI, deliveredPI);
```

在这种情况下，无论消息是被正确发送还是递送失败，都将通过这两个PendingIntent对象来通知您其状态。

8.1.3 使用意图发送SMS消息

使用SmsManager类，可以通过应用程序发送SMS消息，而不需要涉及内置的Messaging应用程序。但有时候，如果可以直接调用内置的Messaging应用程序并让它做发送消息的所有工作，发送SMS消息会变得更容易些。

要在应用程序中激活内置的Messaging应用程序，可以使用一个具有MIME类型vnd.android-dir/mms-sms的Intent对象，如下面的代码片段所示：

```
/** 当活动第一次被创建时调用。 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    btnSendSMS = (Button) findViewById(R.id.btnSendSMS);
    btnSendSMS.setOnClickListener(new View.OnClickListener()
    {
        public void onClick(View v)
        {
            //sendSMS("5556", "Hello my friends!");
            Intent i = new
                Intent(android.content.Intent.ACTION_VIEW);
            i.putExtra("address", "5556; 5558; 5560");

            i.putExtra("sms_body", "Hello my friends!");
            i.setType("vnd.android-dir/mms-sms");
            startActivity(i);
        }
    });
}
```

这将调用Messaging应用程序，如图8-3所示。注意，可以发送SMS给多个收件人，只需要用(在putExtra()方法中)分号分隔每个电话号码就行了。

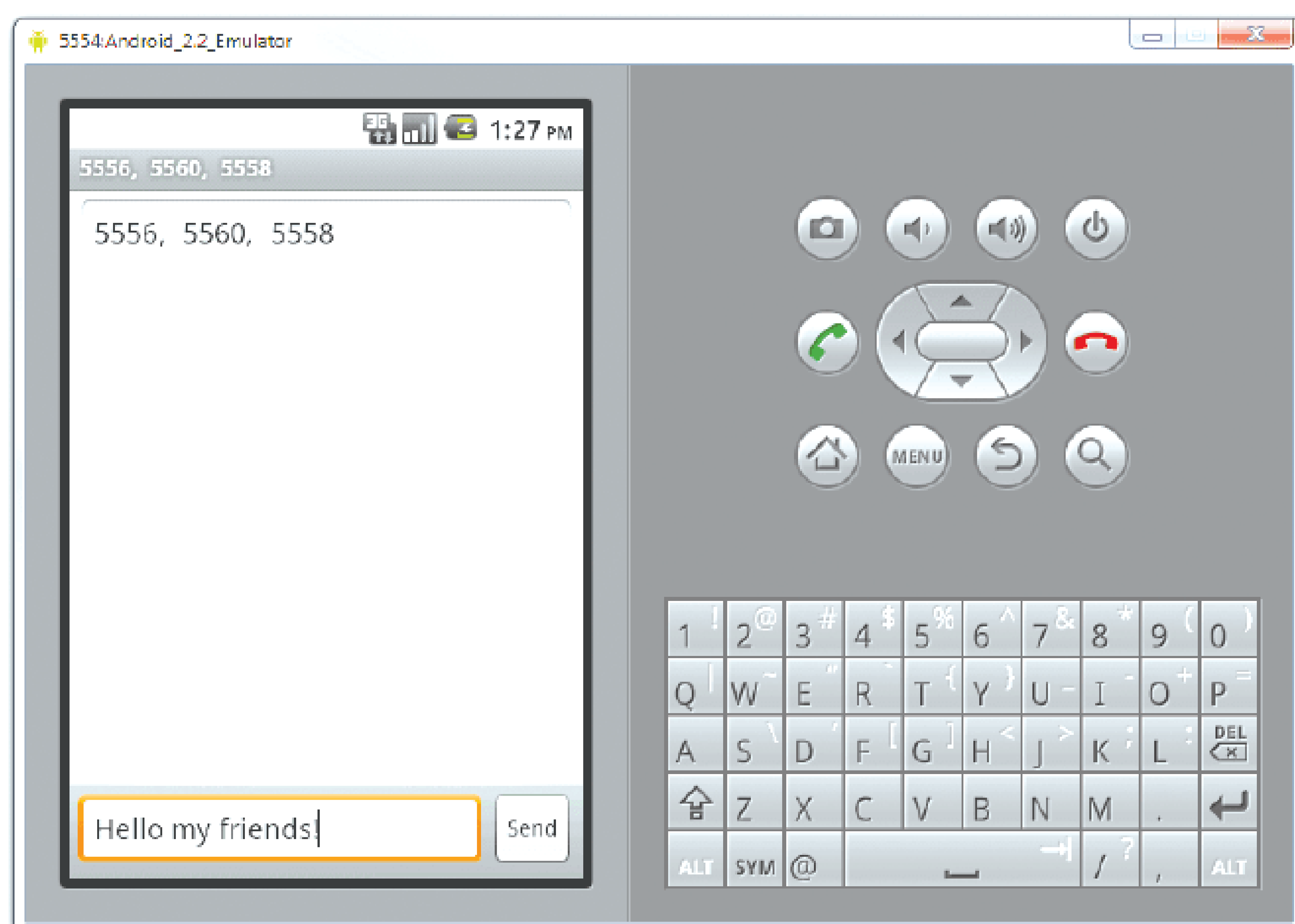
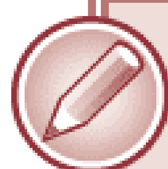


图 8-3



注意： 如果使用这个方法来调用Messaging应用程序，就没有必要在AndroidManifest.xml文件中指定SMS_SEND权限，因为您的应用程序并非是最终发送消息的那个。

8.1.4　接收SMS消息

除了从Android应用程序发送SMS消息外，还可以在应用程序中使用BroadcastReceiver对象接收传入的SMS消息。如果希望应用程序在收到一条特定的SMS消息时执行一个动作，这就很有用了。例如，您可能想追踪您的手机位置以防丢失或被盗。在这种情况下，可以编写一个应用程序，用来自动侦听包含一些秘密代码的SMS消息。一旦收到此类信息，就可以给发送者发回一条包含位置坐标的SMS消息。

下面的“试一试”展示了如何以编程方式侦听传入的SMS消息。

试一试　接收SMS消息

(1) 使用在8.1.3节所创建的同一个项目，在AndroidManifest.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.SMS"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".SMSReceiver">
            <intent-filter>
                <action android:name=
                    "android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
        </receiver>
    </application>
    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
    <uses-permission android:name="android.permission.RECEIVE_SMS">
    </uses-permission>
</manifest>
```

(2) 在项目的src文件夹中，在包名下增加一个新的类文件，并命名为SMSReceiver.java(如图8-4所示)。

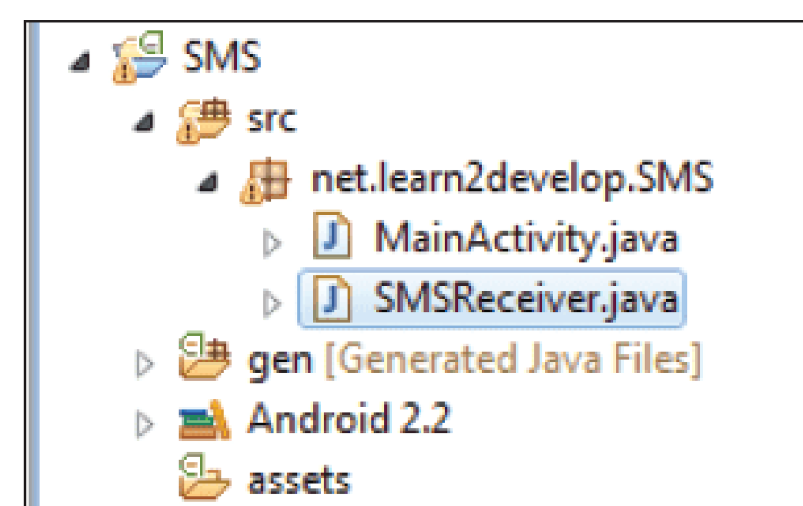


图 8-4

(3) 按如下所示编写SMSReceiver.java文件：

```
package net.learn2develop.SMS;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.widget.Toast;

public class SMSReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        //---获取传入的SMS消息---
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String str = "";
        if (bundle != null)
        {
            //---检索接收到的SMS消息---
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];
            for (int i=0; i<msgs.length; i++){
                msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                str += "SMS from " + msgs[i].getOriginatingAddress();
                str += " :";
                str += msgs[i].getMessageBody().toString();
                str += "\n";
            }
            //---显示新的SMS消息---
            Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
        }
    }
}
```

(4) 按F11键在Android模拟器上调试应用程序。

(5) 使用DDMS，给模拟器发送一条消息。应用程序将能够接收到这条消息并用Toast类进行显示(如图8-5所示)。

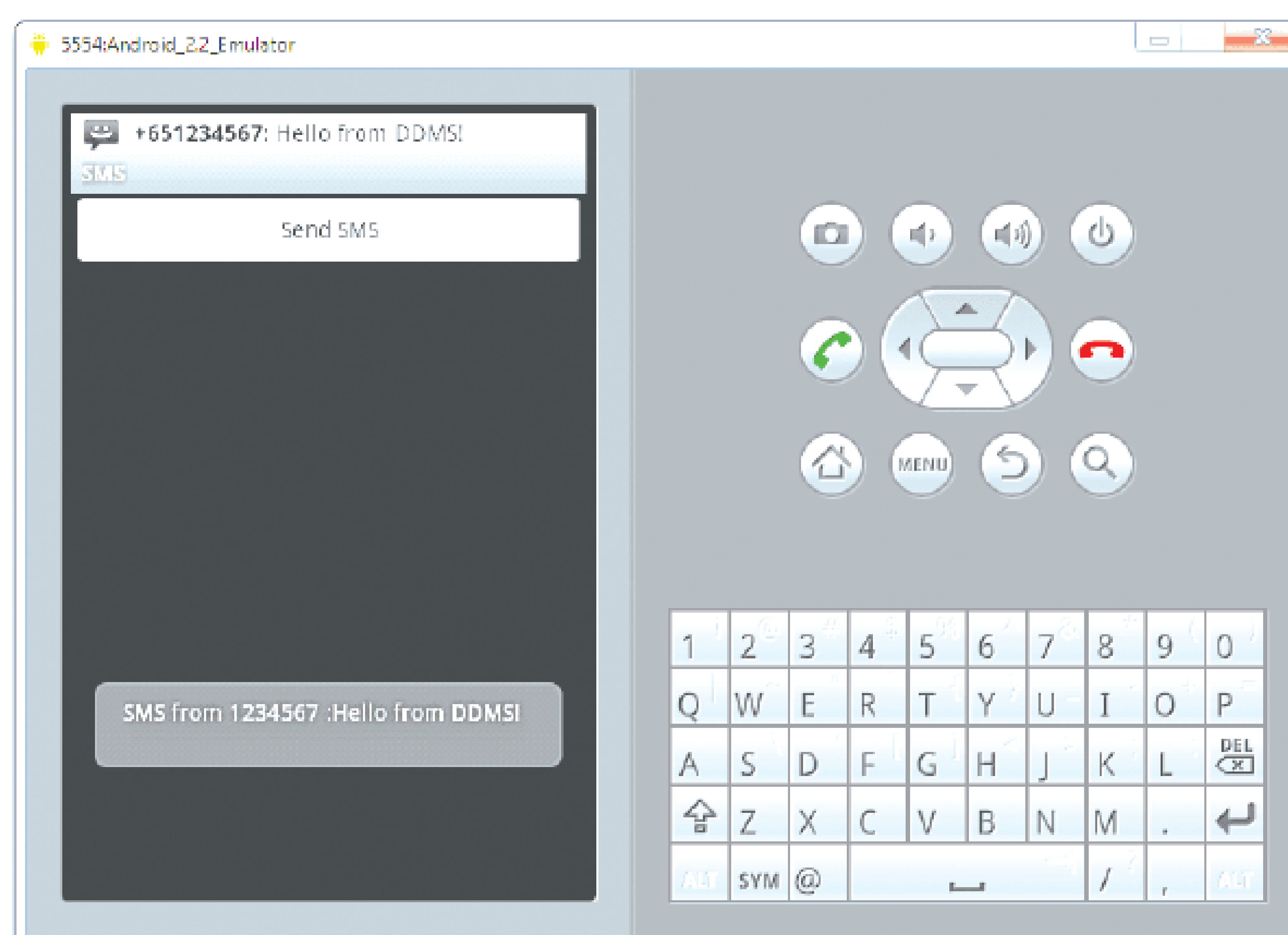


图 8-5

示例说明

要侦听传入的SMS消息，需要创建

一个BroadcastReceiver类。BroadcastReceiver类使应用程序接收其他应用程序使用sendBroadcast()方法发送的意图。从本质上讲，它使您的应用程序可以处理由其他程序应用所引发的事件。当接收到一个意图对象时，调用onReceive()方法。因此，需要重写这一方法。

onReceive()方法在收到一个传入的SMS消息时被触发。SMS消息通过一个Bundle对象包含在Intent对象(即意图；onReceive()方法的第二个参数)中。消息以PDU格式存储在一个Object数组中。要提取每个消息，可以使用SmsMessage类的静态方法createFromPdu()，然后使用Toast类显示SMS消息。发件人的电话号码通过getOriginatingAddress()方法来获得，因此如果需要给发送者发一个自动回复，就可以使用该方法获得发送者的电话号码。

BroadcastReceiver有一个有趣的特性：即使应用程序不在运行，您也可以继续侦听传入的SMS消息；只要应用程序已经安装在设备上，任何传入的SMS消息都将被该应用程序所接收。

1. 通过BroadcastReceiver更新一个活动

上一节介绍了如何使用一个BroadcastReceiver类侦听传入的SMS消息，然后使用Toast类来显示接收到的消息。通常情况下，您想要将SMS消息发回给应用程序的主活动。例如，您可能希望消息在一个TextView中显示。下面的“试一试”将告诉您如何做到这一点。

试一试 创建一个基于视图的应用程序项目

(1) 使用在8.1.3节所创建的同一个项目，在main.xml文件中添加下列粗体显示的行：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
>

<Button
    android:id="@+id/btnSendSMS"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Send SMS" />

<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

</LinearLayout>
```

(2) 在SMSReceiver.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.SMS;

import android.content.BroadcastReceiver;
```

```
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.widget.Toast;

public class SMSReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        //---获取传入的SMS消息---
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String str = "";
        if (bundle != null)
        {
            //---检索接收到的SMS消息---
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];
            for (int i=0; i<msgs.length; i++){
                msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                str += "SMS from " + msgs[i].getOriginatingAddress();
                str += " :";
                str += msgs[i].getMessageBody().toString();
                str += "\n";
            }
            //---显示新的SMS消息---
            Toast.makeText(context, str, Toast.LENGTH_SHORT).show();

            //---发送一个广播意图来更新活动中接收到的SMS---
            Intent broadcastIntent = new Intent();
            broadcastIntent.setAction("SMS_RECEIVED_ACTION");
            broadcastIntent.putExtra("sms", str);
            context.sendBroadcast(broadcastIntent);
        }
    }
}
```

(3) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.SMS;

import android.app.Activity;
import android.os.Bundle;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.telephony.SmsManager;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import android.content.BroadcastReceiver;
import android.content.IntentFilter;
import android.widget.TextView;

public class MainActivity extends Activity {
    Button btnSendSMS;
    IntentFilter intentFilter;

    private BroadcastReceiver intentReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            //---在TextView中显示收到的SMS---
            TextView SMSes = (TextView) findViewById(R.id.textView1);
            SMSes.setText(intent.getExtras().getString("sms"));
        }
    };

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //---用来筛选SMS消息已接收的意图---
        intentFilter = new IntentFilter();
        intentFilter.addAction("SMS_RECEIVED_ACTION");

        btnSendSMS = (Button) findViewById(R.id.btnSendSMS);
        btnSendSMS.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v)
            {
                //sendSMS("5554", "Hello my friends!");

                Intent i = new
                    Intent(android.content.Intent.ACTION_VIEW);
                i.putExtra("address", "5556; 5558; 5560");
                i.putExtra("sms_body", "Hello my friends!");
                i.setType("vnd.android-dir/mms-sms");
                startActivity(i);
            }
        });
    }

    @Override
    protected void onResume() {
```

```
            //---注册接收者---
            registerReceiver(intentReceiver, intentFilter);
            super.onResume();
        }

        @Override
        protected void onPause() {
            //---注销接收者---
            unregisterReceiver(intentReceiver);
            super.onPause();
        }

        //---发送一条SMS消息到另一个设备---
        private void sendSMS(String phoneNumber, String message)
        {
            //...
        }
    }
```

(4) 按F11键在Android模拟器上调试应用程序。使用DDMS向模拟器发送一条SMS消息。图8-6展示了分别由Toast类和TextView所显示的收到的消息。

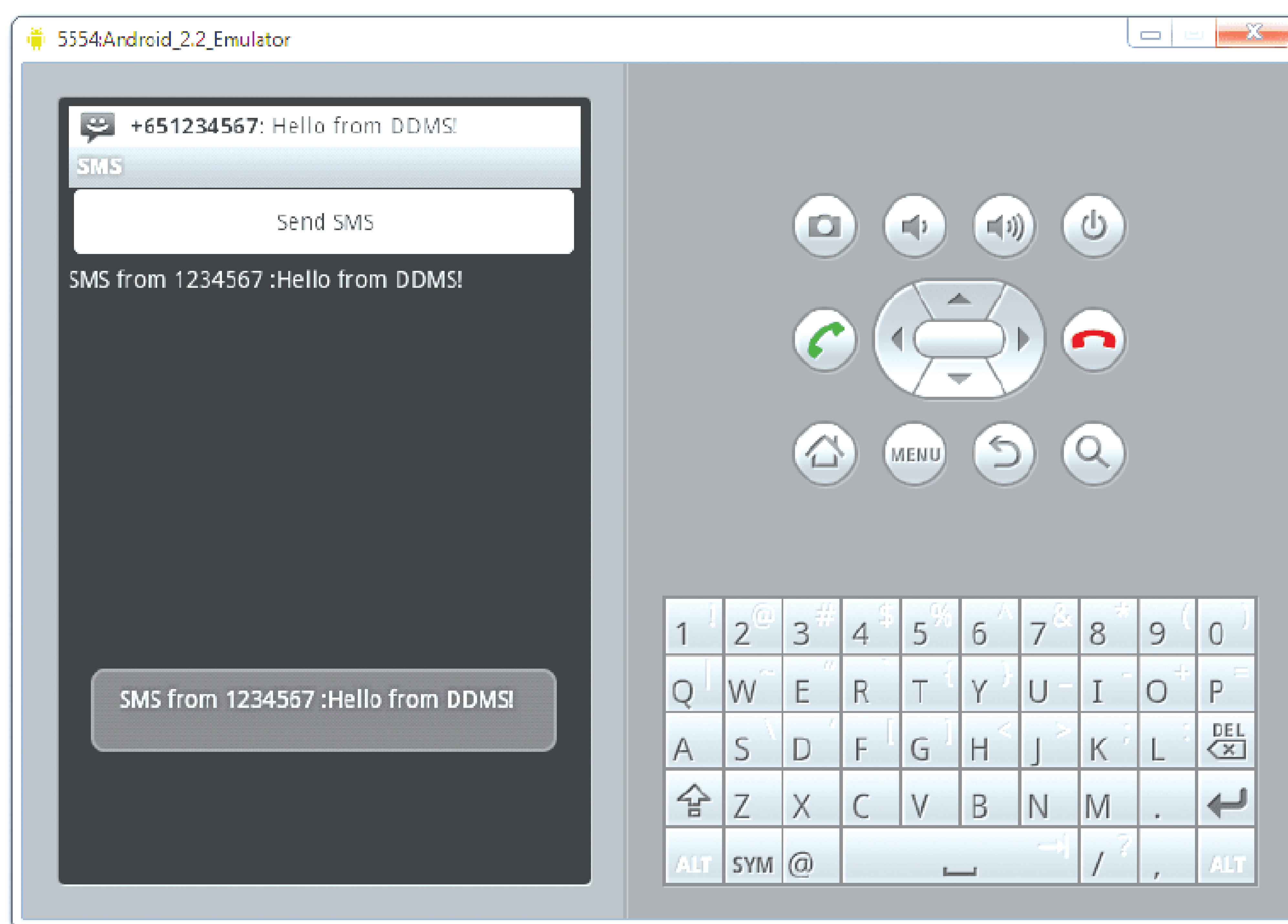


图 8-6

示例说明

首先在活动中添加一个TextView，可以用它来显示接收到的SMS消息。

接下来修改SMSReceiver类。这样当它接收到一条SMS消息时，将广播另一个Intent对象，使得侦听这一意图的任何应用程序可以得到通知(我们接下来将在活动中实现)。收到的SMS也将通过这个意图发送出去。

```
            //---发送一个广播意图来更新活动中接收到的SMS---
```

```
            Intent broadcastIntent = new Intent();
            broadcastIntent.setAction("SMS_RECEIVED_ACTION");
            broadcastIntent.putExtra("sms", str);
            context.sendBroadcast(broadcastIntent);
```

下一步，在活动中创建一个BroadcastReceiver对象来侦听广播意图：

```
private BroadcastReceiver intentReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        //---在TextView中显示收到的SMS---
        TextView SMSes = (TextView) findViewById(R.id.textView1);
        SMSes.setText(intent.getExtras().getString("sms"));
    }
};
```

当收到一个广播意图时，更新TextView中的SMS消息。

您需要创建一个IntentFilter对象以便侦听一个特定的意图。这里，意图是SMS_RECEIVED_ACTION：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    //---用来筛选SMS消息已接收的意图---
    intentFilter = new IntentFilter();
    intentFilter.addAction("SMS_RECEIVED_ACTION");
    //...
}
```

最后，在活动的onResume()事件中注册BroadcastReceiver，在onPause()事件中进行注销：

```
@Override
protected void onResume() {
    //---注册接收者---
    registerReceiver(intentReceiver, intentFilter);
    super.onResume();
}

@Override
protected void onPause() {
    //---注销接收者---
    unregisterReceiver(intentReceiver);
    super.onPause();
}
```

这意味着，只有在收到消息而活动在屏幕上可见时，TextView才会显示该消息。如果接收到SMS消息时活动不在前台，TextView将不会被更新。

2. 通过BroadcastReceiver调用一个活动

前面的示例说明了如何传递接收到的SMS消息来在活动中显示。然而，在许多情况下，当接收SMS消息时活动可能在后台。在这种情况下，当接收一条消息时，如果能将活动推到前台将是很有用的。下面的“试一试”展示了该如何做到这一点。

试一试 调用一个活动

(1) 使用在8.1.3节所创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的行：

```
/** 当活动第一次被创建时调用。 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    //---用来筛选SMS消息已接收的意图---
    intentFilter = new IntentFilter();
    intentFilter.addAction("SMS_RECEIVED_ACTION");

    //---注册接收者---
    registerReceiver(intentReceiver, intentFilter);

    btnSendSMS = (Button) findViewById(R.id.btnSendSMS);
    btnSendSMS.setOnClickListener(new View.OnClickListener()
    {
        public void onClick(View v)
        {
            //sendSMS("5554", "Hello my friends!");
            Intent i = new
                Intent(android.content.Intent.ACTION_VIEW);
            i.putExtra("address", "5556; 5558; 5560");


            i.putExtra("sms_body", "Hello my friends!");
            i.setType("vnd.android-dir/mms-sms");
            startActivity(i);
        }
    });
}

@Override
protected void onResume() {
    //---注册接收者---
    //registerReceiver(intentReceiver, intentFilter);
    super.onResume();
}

@Override
```

```
protected void onPause() {
    //---注销接收者---
    //unregisterReceiver(intentReceiver);
    super.onPause();
}

@Override
protected void onDestroy() {
    //---注销接收者---
    unregisterReceiver(intentReceiver);
    super.onPause();
}
```

(2) 在SMSReceiver.java文件中添加下列粗体显示的语句：

```
@Override
public void onReceive(Context context, Intent intent)
{
    //---获取传入的SMS消息---
    Bundle bundle = intent.getExtras();
    SmsMessage[] msgs = null;
    String str = "";
    if (bundle != null)
    {
        //---检索接收到的SMS消息---
        Object[] pdus = (Object[]) bundle.get("pdus");
        msgs = new SmsMessage[pdus.length];
        for (int i=0; i<msgs.length; i++){
            msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
            str += "SMS from " + msgs[i].getOriginatingAddress();
            str += " :";
            str += msgs[i].getMessageBody().toString();
            str += "\n";
        }
        //---显示新的SMS消息---
        Toast.makeText(context, str, Toast.LENGTH_SHORT).show();

        //---启动MainActivity---
        Intent mainActivityIntent = new Intent(context, MainActivity.class);
        mainActivityIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(mainActivityIntent);

        //---发送一个广播意图来更新活动中接收到的SMS---
        Intent broadcastIntent = new Intent();
        broadcastIntent.setAction("SMS_RECEIVED_ACTION");
        broadcastIntent.putExtra("sms", str);
        context.sendBroadcast(broadcastIntent);
    }
}
```

(3) 按如下所示修改main.xml文件：

```
<activity android:name=".MainActivity"
          android:label="@string/app_name"
          android:launchMode="singleTask" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

(4) 按F11键在Android模拟器上调试应用程序。当MainActivity显示时，单击Home按钮将活动发送到后台。

(4) 使用DDMS再次向模拟器发送一条SMS消息。这一次，注意活动将被推到前台，显示接收的SMS消息。

示例说明

在MainActivity类中，首先在活动的OnCreate()事件而不是onResume()事件中注册BroadcastReceiver，并在onDestroy()事件而不是onPause()事件中进行注销。这确保了活动即使是在后台，它仍能够侦听广播意图。

接下来，修改SMSReceiver类中的onReceive()事件，使用一个意图在广播另一个意图之前将活动推到前台：

```
//---启动MainActivity---
Intent mainActivityIntent = new Intent(context, MainActivity.class);
mainActivityIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
context.startActivity(mainActivityIntent);

//---发送一个广播意图来更新活动中接收到的SMS---
Intent broadcastIntent = new Intent();
broadcastIntent.setAction("SMS_RECEIVED_ACTION");
broadcastIntent.putExtra("sms", str);
context.sendBroadcast(broadcastIntent);
```

startActivity()方法启动活动，并将它推到前台。注意，需要设置Intent.FLAG_ACTIVITY_NEW_TASK标志，因为从一个活动上下文的外部调用startActivity()需要FLAG_ACTIVITY_NEW_TASK标志。

还需要将AndroidManifest.xml文件中<activity>元素的launchMode属性设置为 singleTask：

```
<activity android:name=".MainActivity"
          android:label="@string/app_name"
          android:launchMode="singleTask" >
```

如果不进行设置，在应用程序收到SMS消息时，将启动活动的多个实例。

注意在这个示例中，当活动在后台时(如单击Home按钮来显示主屏幕)，随着SMS消息的接收，活动将被推到前台并且TextView得到更新。但是，如果活动被终止(如单击Back按钮来销毁它)，活动会再次启动，但TextView不会被更新。

8.1.5　说明和警告

虽然发送和接收SMS消息的能力使Android成为开发复杂应用程序的一个非常引人注目的平台，但这种灵活性是要付出代价的。貌似无害的应用程序可能在幕后发送SMS消息而用户对此一无所知。正如最近一个基于SMS的Android木马程序的例子(http://forum.vodafone.co.nz/topic/5719-android-sms-trojan-warning/)，该应用程序自称是一个媒体播放器，一旦安装，它将向一个收费昂贵的号码发送SMS消息，导致用户产生巨额话费。

尽管用户需要显式地将权限授予应用程序，但也仅仅是在安装时才显示对权限的请求。图8-7展示了在模拟器上(与真正的设备上相同)试图安装应用程序(作为一个APK文件；第11章将对如何打包Android应用程序进行详细讨论)时所显示的权限请求。如果用户单击Install按钮，他或她将被认为授予了权限，允许应用程序发送和接收SMS消息。这是很危险的，因为在应用程序安装后，它可以发送和接收SMS消息，而不再给用户任何提示。

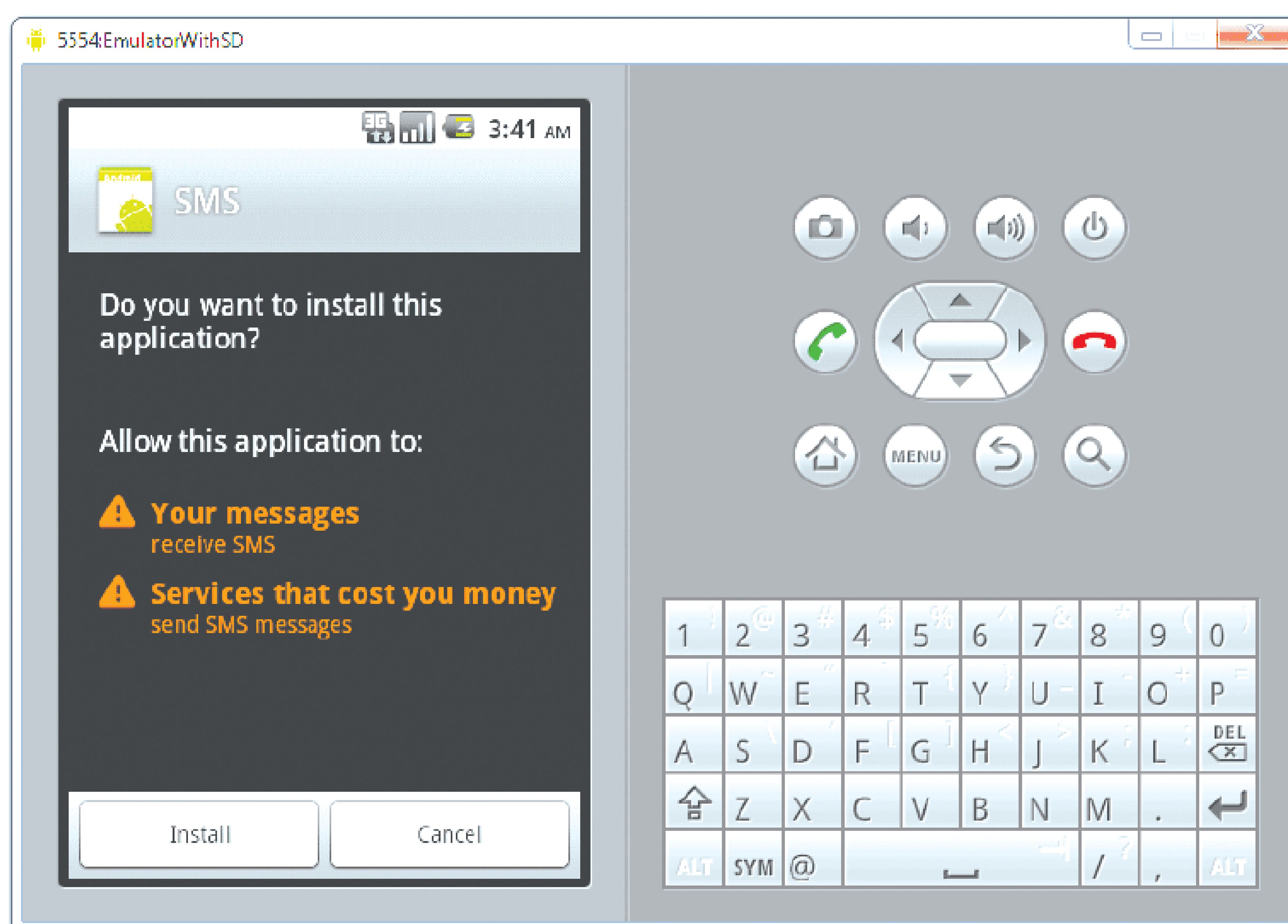


图 8-7

此外，应用程序还可以“嗅探”传入的SMS消息。例如，在8.1.4节所学到的技术基础上，可以很容易地编写出检查SMS消息中某些关键字的应用程序。当SMS消息包含正在查找的关键字时，可以使用Location Manager(将在第9章中讨论)来获取您的地理位置，然后给SMS消息的发送者发回坐标。这样，这个发送者就可以很容易地追踪您的位置。所有这些任务都可以很容易地在您一无所知的情况下完成！也就是说，用户应该尽量避免安装来历不明(例如来自未知的网站、陌生人等)的Android应用程序。

8.2　发送电子邮件

与SMS消息传递类似，Android还支持电子邮件。Android上的Gmail/Email应用程序可以使您使用POP3或IMAP来配置电子邮件账户。除了使用Gmail/Email应用程序发送和接收电子邮件外，还可以通过编程方式从Android应用程序中发送电子邮件。下面的“试一试”将告诉您如何做。

试一试　以编程方式发送电子邮件

Emails.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个名为Emails的新的Android项目。

(2) 在main.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

<Button
    android:id="@+id/btnSendEmail"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Send Email" />

</LinearLayout>
```

(3) 在MainActivtiy.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Email;

import android.app.Activity;
import android.os.Bundle;

import android.content.Intent;
import android.net.Uri;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {
    Button btnSendEmail;

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        btnSendEmail = (Button) findViewById(R.id.btnSendEmail);
        btnSendEmail.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v)
            {
                String[] to = {"weimenglee@learn2develop.net",
                               "weimenglee@gmail.com"};
```

```
                String[] cc = {"course@learn2develop.net"};
                sendEmail(to, cc, "Hello", "Hello my friends!");
            }
        });
    }

    //---发送一条SMS消息到另一个设备---
    private void sendEmail(String[] emailAddresses, String[] carbonCopies,
    String subject, String message)
    {
        Intent emailIntent = new Intent(Intent.ACTION_SEND);
        emailIntent.setData(Uri.parse("mailto:"));
        String[] to = emailAddresses;
        String[] cc = carbonCopies;
        emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
        emailIntent.putExtra(Intent.EXTRA_CC, cc);
        emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
        emailIntent.putExtra(Intent.EXTRA_TEXT, message);
        emailIntent.setType("message/rfc822");
        startActivity(Intent.createChooser(emailIntent, "Email"));
    }
}
```

(4) 按F11键在真正的Android设备上调试应用程序。单击Send Email按钮可以看到在设备上启动了Email应用程序，如图8-8所示。

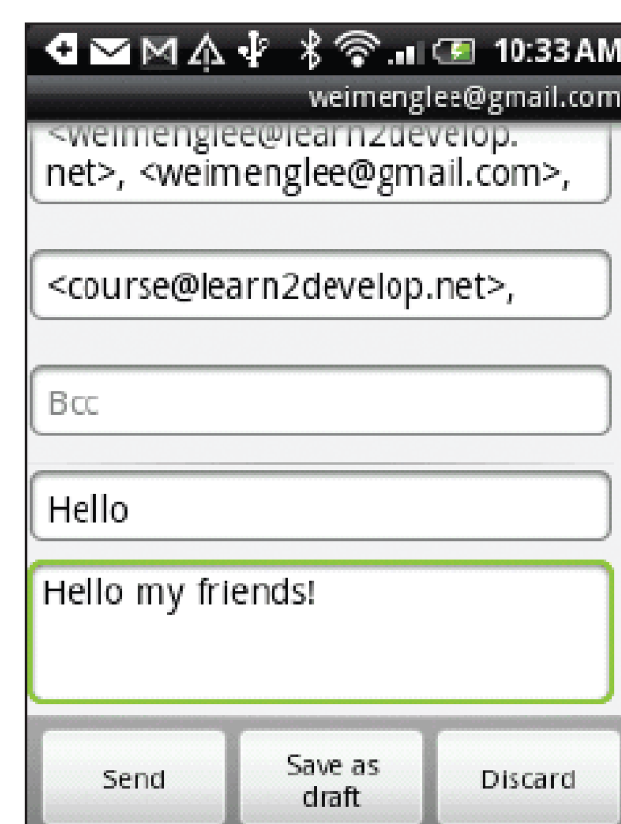


图 8-8

示例说明

在这个示例中，启动内置的Email应用程序来发送一封电子邮件消息。要做到这一点，可以使用一个Intent对象并使用setData()、putExtra()和setType()方法来设置各种参数：

```
        Intent emailIntent = new Intent(Intent.ACTION_SEND);
        emailIntent.setData(Uri.parse("mailto:"));
        String[] to = emailAddresses;
        String[] cc = carbonCopies;
        emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
        emailIntent.putExtra(Intent.EXTRA_CC, cc);
        emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
        emailIntent.putExtra(Intent.EXTRA_TEXT, message);
        emailIntent.setType("message/rfc822");
        startActivity(Intent.createChooser(emailIntent, "Email"));
```

8.3　联网

前面的章节讲述了如何利用SMS和电子邮件与外界进行连接。实现这一目标的另一种方式是使用HTTP协议。使用HTTP协议，可以执行各种任务，如从Web服务器上下载Web页面、下载二进制数据等。

下面的“试一试”创建了一个新的Android项目，可以使用HTTP协议连接到Web上来下载各种数据。

试一试 创建项目

Networking.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，创建一个名为Networking的新的Android项目。

(2) 在AndroidManifest.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Networking"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.INTERNET"> </uses-permission>
</manifest>
```

(3) 在MainActivity.java文件中导入以下命名空间：

```
package net.learn2develop.Networking;

import android.app.Activity;
import android.os.Bundle;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.widget.ImageView;
import android.widget.Toast;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
```

```
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

(4) 在MainActivity.java文件中定义OpenHttpConnection()方法：

```
public class MainActivity extends Activity {

    private InputStream OpenHttpConnection(String urlString)
    throws IOException
    {
        InputStream in = null;
        int response = -1;

        URL url = new URL(urlString);
        URLConnection conn = url.openConnection();

        if (!(conn instanceof HttpURLConnection))
            throw new IOException("Not an HTTP connection");
        try{
            HttpURLConnection httpConn = (HttpURLConnection) conn;
            httpConn.setAllowUserInteraction(false);
            httpConn.setInstanceFollowRedirects(true);
            httpConn.setRequestMethod("GET");
            httpConn.connect();
            response = httpConn.getResponseCode();
            if (response == HttpURLConnection.HTTP_OK) {
                in = httpConn.getInputStream();
            }
        }
        catch (Exception ex)
        {
            throw new IOException("Error connecting");
        }
        return in;
    }

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
        }
    }
```

示例说明

由于是使用HTTP协议连接到Web上，所以应用程序需要INTERNET权限。因此，首先要做的是在AndroidManifest.xml文件中添加这一权限。

然后定义OpenHttpConnection()方法，它接受一个URL字符串，并返回一个InputStream对象。使用InputStream对象，可以从流对象中读取字节以下载数据。在此方法中，利用HttpURLConnection对象，打开一个到远程URL的HTTP连接。设置连接的各种属性，如请求方法等：

```
HttpURLConnection httpConn = (HttpURLConnection) conn;
httpConn.setAllowUserInteraction(false);
httpConn.setInstanceFollowRedirects(true);
httpConn.setRequestMethod("GET");
```

在尝试建立与服务器的连接后，从服务器获得HTTP响应码。如果建立了连接(通过响应码HTTP_OK)，那么就可以继续从连接获取一个InputStream对象：

```
httpConn.connect();
response = httpConn.getResponseCode();
if (response == HttpURLConnection.HTTP_OK) {
    in = httpConn.getInputStream();
}
```

使用InputStream对象，就可以开始从服务器上下载数据了。

8.3.1 下载二进制数据

需要执行的常见任务之一是从Web上下载二进制数据。例如，您可能想从一台服务器上下载一幅图像以在应用程序中显示。下面的“试一试”说明了这一任务是如何完成的。

试一试 创建项目

(1) 打开先前创建的同一个项目，在main.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

<ImageView
```

```
        android:id="@+id/img"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />

</LinearLayout>
```

(2) 在MainActivity.java文件中添加下列粗体显示的语句：

```
public class MainActivity extends Activity {
    ImageView img;

    private InputStream OpenHttpConnection(String urlString)
    throws IOException
    {
        //...
    }

    private Bitmap DownloadImage(String URL)
    {
        Bitmap bitmap = null;
        InputStream in = null;
        try {
            in = OpenHttpConnection(URL);
            bitmap = BitmapFactory.decodeStream(in);
            in.close();
        } catch (IOException e1) {
            Toast.makeText(this, e1.getLocalizedMessage(),
                Toast.LENGTH_LONG).show();

            e1.printStackTrace();
        }
        return bitmap;
    }

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //---下载一幅图像---
        Bitmap bitmap =
            DownloadImage(
            "http://www.streetcar.org/mim/cable/images/cable-01.jpg");
        img = (ImageView) findViewById(R.id.img);
        img.setImageBitmap(bitmap);
    }
}
```

(3) 按F11键在Android模拟器上调试应用程序。图8-9展示了从Web上下载了一幅图像并在ImageView中显示。

示例说明

DownloadImage()方法接受要下载的图像的URL，然后使用之前定义过的OpenHttp-Connection()方法打开到服务器的连接。利用连接返回的InputStream对象，使用BitmapFactory类的decodeStream()方法来下载和解码数据，使之成为一个Bitmap对象。DownloadImage()方法返回一个Bitmap对象。



图 8-9

然后，使用一个ImageView视图将该图像显示出来。

从模拟器指向localhost

使用Android模拟器工作时，可能经常要使用localhost来访问驻留在本地Web服务器上的数据。例如，您自己的Web服务很可能在开发期间驻留在本地的计算机上，而且您希望在用于编写Android应用程序的同一台开发机上测试它。在这种情况下，应该使用特殊的IP地址10.0.2.2（而不是127.0.0.1)来指向主机的回环接口。从Android模拟器的角度来看，localhost (127.0.0.1)指的是模拟器自己的回环接口。

8.3.2 下载文本文件

除了下载二进制数据，还可以下载纯文本文件。例如，您可能会写一个RSS Reader应用程序，因此需要下载RSS XML源来进行处理。下面的“试一试”说明了如何在应用程序中下载纯

文本文件。

试一试　下载纯文本文件

(1) 打开先前创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
public class MainActivity extends Activity {
    ImageView img;

    private InputStream OpenHttpConnection(String urlString)
    throws IOException
    {
        //...
    }

    private Bitmap DownloadImage(String URL)
    {
        //...
    }

    private String DownloadText(String URL)
    {
        int BUFFER_SIZE = 2000;
        InputStream in = null;
        try {
            in = OpenHttpConnection(URL);
        } catch (IOException e1) {
            Toast.makeText(this, e1.getLocalizedMessage(),
                Toast.LENGTH_LONG).show();

            e1.printStackTrace();
            return "";
        }

        InputStreamReader isr = new InputStreamReader(in);
        int charRead;
        String str = "";
        char[] inputBuffer = new char[BUFFER_SIZE];
        try {
            while ((charRead = isr.read(inputBuffer))>0)
            {
                //---将字符转换成字符串---
                String readString =
                    String.copyValueOf(inputBuffer, 0, charRead);
                str += readString;
                inputBuffer = new char[BUFFER_SIZE];
            }
            in.close();
        } catch (IOException e) {
            Toast.makeText(this, e.getLocalizedMessage(),
                Toast.LENGTH_LONG).show();
```

```
                e.printStackTrace();
                return "";
            }
            return str;
        }

        /**当活动第一次被创建时调用。 */
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);

            //---下载一副图像---
            Bitmap bitmap =
                DownloadImage(
                "http://www.streetcar.org/mim/cable/images/cable-01.jpg");
            img = (ImageView) findViewById(R.id.img);
            img.setImageBitmap(bitmap);

            //---下载一个RSS源---
            String str = DownloadText(
                "http://www.appleinsider.com/appleinsider.rss");
            Toast.makeText(getBaseContext(), str,
                           Toast.LENGTH_SHORT).show();
        }
    }
```

(2) 按F11键在Android模拟器上调试应用程序。图8-10展示了下载RSS源并用Toast类进行显示。



图 8-10

示例说明

DownloadText()方法接受要下载的文本文件的URL，然后返回下载的文本文件的字符串。它

基本上是打开一个到服务器的HTTP连接，然后使用一个InputStreamReader对象从流中读取每个字符，将它们保存在一个String对象中。

8.3.3 访问Web服务

到目前为止，您已经了解了如何从Web上下载图像和文本。8.3.2节展示了如何从服务器下载一个RSS源。很多时候，您需要下载XML文件并解析其内容(使用Web服务就是这样一个很好的示例)。因此，在本节中将学习如何使用HTTP的GET方法连接到一个Web服务。一旦Web服务返回一个XML格式的结果，就可提取相关部分，并使用Toast类显示其内容。

在这个示例中，将要使用的Web方法来自http://services.aonaware.com/DictService/DictService.asmx?op=Define。此Web方法是从Dictionary这个Web服务返回一个给定单词的定义。

Web方法按以下格式接受一个请求：

```
GET /DictService/DictService.asmx/Define?word=string HTTP/1.1
Host: services.aonaware.com
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

它按如下格式返回响应：

```
<?xml version="1.0" encoding="utf-8"?>
<WordDefinition xmlns="http://services.aonaware.com/webservices/">
  <Word>string</Word>
  <Definitions>
    <Definition>
      <Word>string</Word>
      <Dictionary>
        <Id>string</Id>
        <Name>string</Name>
      </Dictionary>
      <WordDefinition>string</WordDefinition>
    </Definition>
    <Definition>
      <Word>string</Word>
      <Dictionary>
        <Id>string</Id>
        <Name>string</Name>
      </Dictionary>
      <WordDefinition>string</WordDefinition>
    </Definition>
  </Definitions>
</WordDefinition>
```

因此，要获得一个单词的定义，需要建立一个到Web方法的HTTP连接，然后解析返回的XML结果。下面的“试一试”将告诉您该如何做。

试一试 使用Web服务

(1) 使用先前创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
public class MainActivity extends Activity {
    ImageView img;

    private InputStream OpenHttpConnection(String urlString)
    throws IOException
    {
        //...
    }

    private Bitmap DownloadImage(String URL)
    {
        //...
    }

    private String DownloadText(String URL)
    {
        //...
    }

    private void WordDefinition(String word) {
        InputStream in = null;
        try {
            in = OpenHttpConnection(
"http://services.aonaware.com/DictService/DictService.asmx/Define?word=" + word);
            Document doc = null;
            DocumentBuilderFactory dbf =
                DocumentBuilderFactory.newInstance();
            DocumentBuilder db;
            try {
                db = dbf.newDocumentBuilder();
                doc = db.parse(in);
            } catch (ParserConfigurationException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            doc.getDocumentElement().normalize();

            //---检索所有的<Definition>节点---
            NodeList itemNodes =
                doc.getElementsByTagName("Definition");

            String strDefinition = "";
            for (int i = 0; i < definitionElements.getLength(); i++) {
```

```
                    Node itemNode = definitionElements.item(i);
                    if (itemNode.getNodeType() == Node.ELEMENT_NODE)
                    {
                        //---将节点转换为元素---
                        Element definitionElement = (Element) itemNode;

                        //---获取在<Definition>元素下的所有<WordDefinition>元素---
                        NodeList wordDefinitionElements =
                            (definitionElement).getElementsByTagName(
                            "WordDefinition");

                        strDefinition = "";
                        for (int j = 0; j < wordDefinitionElements.getLength();
                            j++) {
                            //---将<WordDefinition>节点转换为元素---
                            Element wordDefinitionElement =
                                (Element) wordDefinitionElements.item(j);

                            //---获取在<WordDefinition>元素下的所有子节点---
                            NodeList textNodes =
                                ((Node) wordDefinitionElement).getChildNodes();

                            strDefinition +=
                                ((Node) textNodes.item(0)).getNodeValue() + ". ";
                        }

                        //---显示标题---
                        Toast.makeText(getBaseContext(),strDefinition,
                            Toast.LENGTH_SHORT).show();
                    }
                }
            } catch (IOException e1) {
                Toast.makeText(this, e1.getLocalizedMessage(),
                    Toast.LENGTH_LONG).show();
                e1.printStackTrace();
            }
        }

        /** 当活动第一次被创建时调用。 */
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);

            //---下载一幅图像---
            Bitmap bitmap =
                DownloadImage(
                "http://www.streetcar.org/mim/cable/images/cable-01.jpg");
            img = (ImageView) findViewById(R.id.img);
            img.setImageBitmap(bitmap);
```

```
            //---下载一个RSS源---
            String str = DownloadText(
                "http://www.appleinsider.com/appleinsider.rss");
            Toast.makeText(getBaseContext(), str,
                           Toast.LENGTH_SHORT).show();

            //---使用GET访问Web服务---
            WordDefinition("Apple");
        }
    }
```

(2) 按F11键在Android模拟器上调试应用程序。图8-11展示了解析Web服务调用的结果并用Toast类进行显示。

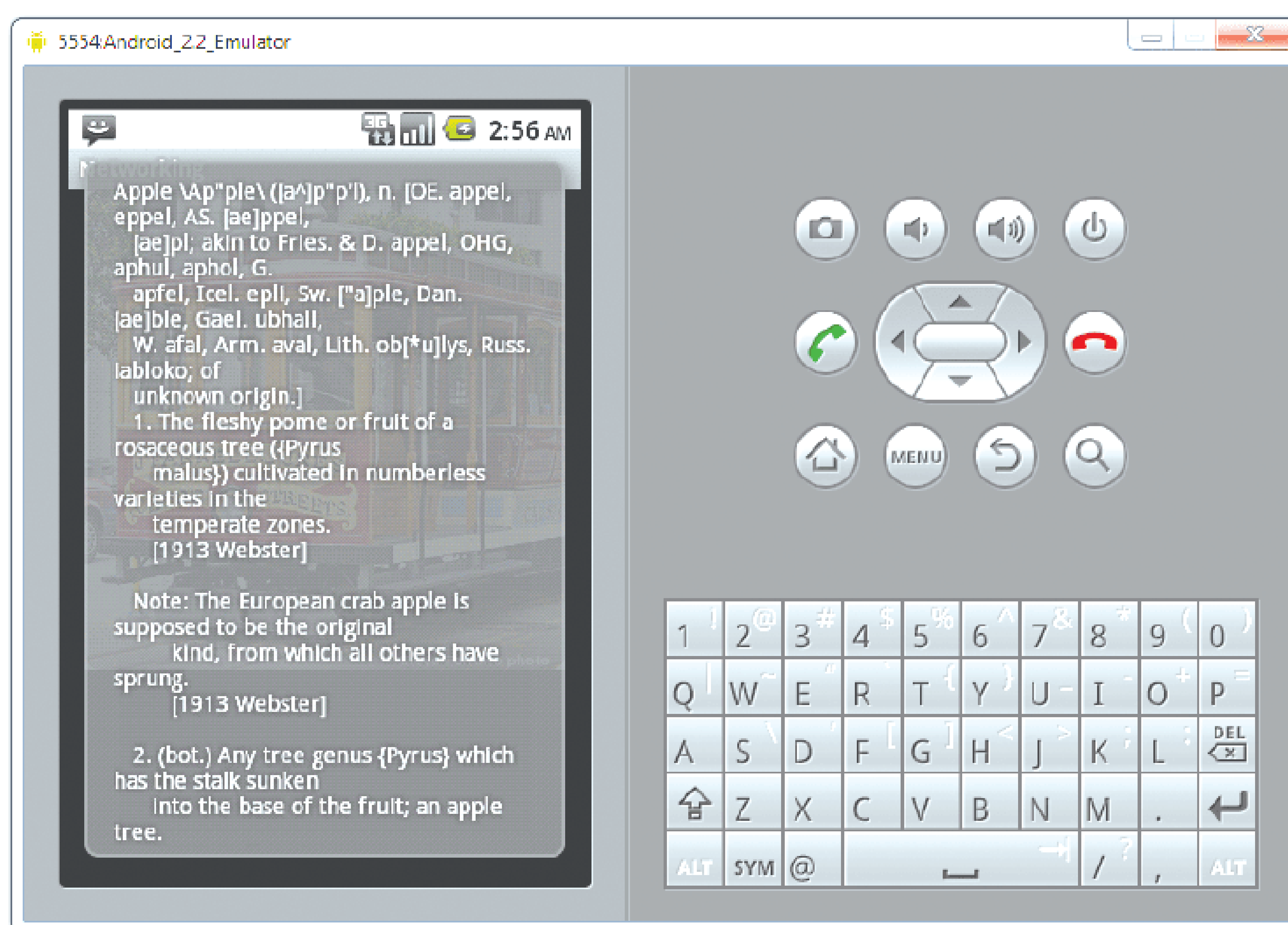


图 8-11

示例说明

WordDefinition()方法首先打开一个到Web服务的HTTP连接，传入一个您感兴趣的单词：

```
                in = OpenHttpConnection(
    "http://services.aonaware.com/DictService/DictService.asmx/Define?word=" + word);
```

然后，它使用DocumentBuilderFactory和DocumentBuilder对象从XML文件(这一文件是由Web服务返回的XML结果)中获得一个Document对象(DOM)：

```
                Document doc = null;
                DocumentBuilderFactory dbf =
                    DocumentBuilderFactory.newInstance();
                DocumentBuilder db;
                try {
                    db = dbf.newDocumentBuilder();
```

```
        doc = db.parse(in);
    } catch (ParserConfigurationException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    doc.getDocumentElement().normalize();
```

一旦获得Document对象，您会发现所有具有<Definition>标记的元素：

```
    //---检索所有的<Definition>节点---
    NodeList itemNodes =
        doc.getElementsByTagName("Definition");
```

图8-12展示了由Web服务返回的XML文档的结构。

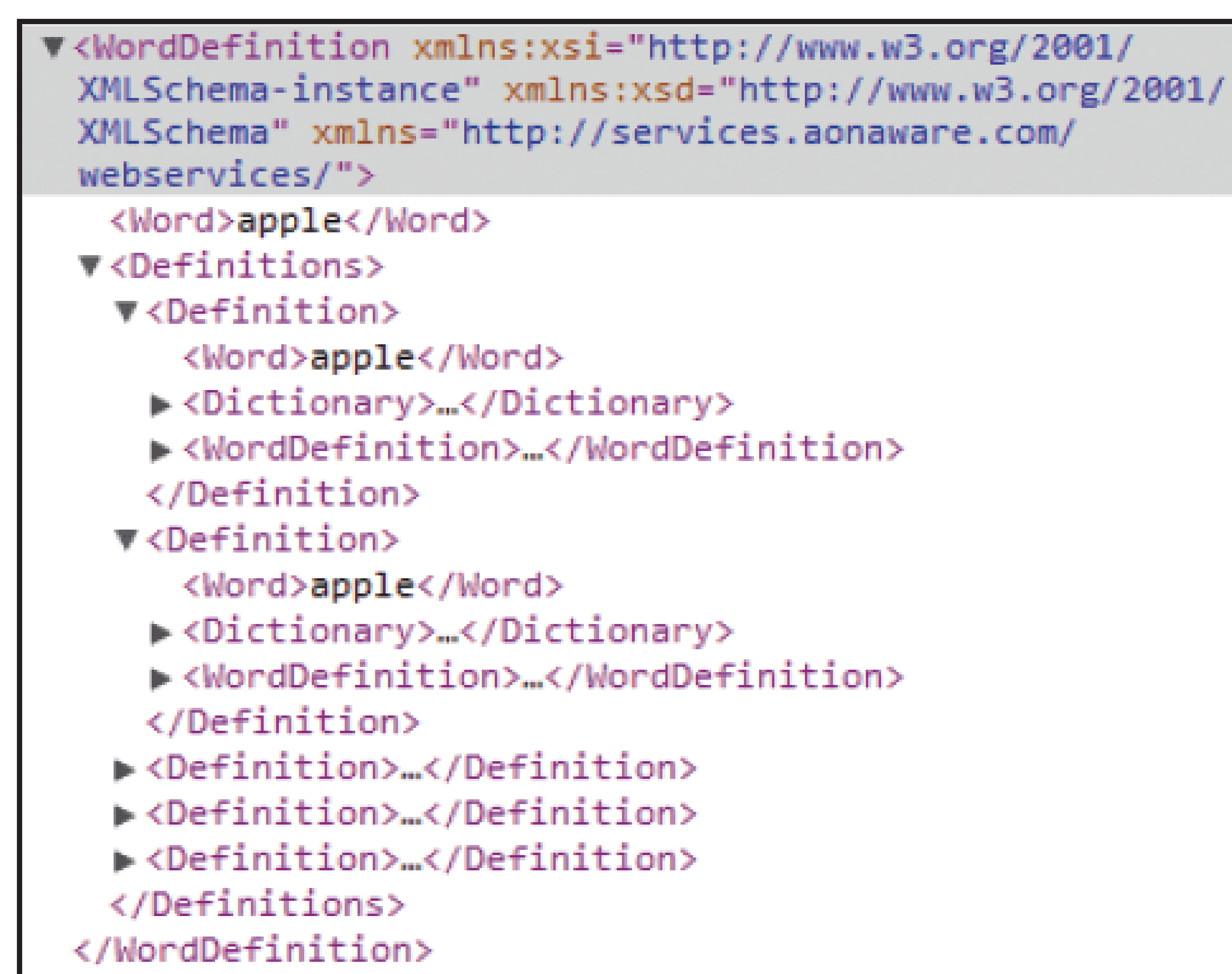


图 8-12

由于单词的定义包含在<WordDefinition>元素中，因此继续对所有的定义进行提取：

```
    String strDefinition = "";
    for (int i = 0; i < definitionElements.getLength(); i++) {
        Node itemNode = definitionElements.item(i);
        if (itemNode.getNodeType() == Node.ELEMENT_NODE)
        {
            //---将节点转换为元素---
            Element definitionElement = (Element) itemNode;

            //---获取在<Definition>元素下的所有<WordDefinition>元素---
            NodeList wordDefinitionElements =
                (definitionElement).getElementsByTagName(
                "WordDefinition");

            strDefinition = "";
            for (int j = 0; j < wordDefinitionElements.getLength();
                j++) {
                //---将<WordDefinition>节点转换为元素---
```

```
                    Element wordDefinitionElement =
                        (Element) wordDefinitionElements.item(j);

                    //---获取在<WordDefinition>元素下的所有子节点---
                    NodeList textNodes =
                        ((Node) wordDefinitionElement).getChildNodes();
                    //---获取第一个包含文本的节点---
                    strDefinition +=
                        ((Node) textNodes.item(0)).getNodeValue() + ". ";
                }
                //---显示标题---
                Toast.makeText(getBaseContext(),strDefinition,
                    Toast.LENGTH_SHORT).show();
            }
        }
    } catch (IOException e1) {
        Toast.makeText(this, e1.getLocalizedMessage(),
            Toast.LENGTH_LONG).show();
        e1.printStackTrace();
    }
```

上述代码循环遍历所有<Definition>元素，然后为每一个<Definition>元素寻找名为<WordDefinition>的子元素。单词的定义包含在<WordDefinition>元素的文本内容中。Toast类显示检索到的每一个单词的定义。

8.3.4 执行异步调用

到目前为止，在前面几节中建立的所有连接都是同步的——即与服务器的连接直到接收了数据后才返回。在现实生活中，由于网络连接固有的缓慢的特征，这样做存在一些问题。当连接到一个服务器来下载一些数据时，应用程序的用户界面在没有获得响应之前将保持“冻结”的状态。在大多数情况下，这是不能被接受的。因此，需要确保与服务器的连接是按异步的方式进行。

异步连接到服务器的最容易的方式是使用Android SDK提供的AsyncTask类。使用AsyncTask，可以使您在一个单独的线程中执行后台任务，并在一个UI线程中返回结果。使用这个类可以使您执行后台操作，而无须处理复杂的线程问题。

使用先前从服务器上下载一幅图像，然后显示在ImageView中的示例，可以在AsyncTask类的一个实例中包含如下所示代码：

```
public class MainActivity extends Activity {
    ImageView img;

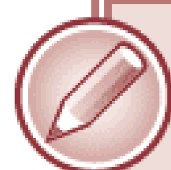
private class BackgroundTask extends AsyncTask
    <String, Void, Bitmap> {
        protected Bitmap doInBackground(String... url) {
            //---下载一幅图像---
            Bitmap bitmap = DownloadImage(url[0]);
```

```
            return bitmap;
        }

        protected void onPostExecute(Bitmap bitmap) {
            ImageView img = (ImageView) findViewById(R.id.img);
            img.setImageBitmap(bitmap);
        }
    }

    private InputStream OpenHttpConnection(String urlString)
    throws IOException
    {
        ...
    }
```

这基本上就是定义了一个扩展AsyncTask类的类。这样，在BackgroundTask类中就有两个方法——doInBackground()和onPostExecute()。将所有需要异步运行的代码放在doInBackground()方法中。当任务完成后，其结果通过onPostExecute()方法传回。 onPostExecute()方法是在UI线程上执行的，因此使用从服务器上下载的位图来更新ImageView是线程安全的。



注意：在第10章中将学习更多关于AsyncTask类的内容，这一章涵盖了如何在Android中开发服务。

要执行异步任务，只需要创建BackgroundTask类的一个实例，并调用它的execute()方法：

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
            new BackgroundTask().execute(
                "http://www.streetcar.org/mim/cable/images/cable-01.jpg");
}
```

8.4　本章小结

本章讲述了与外界通信的各种方式。首先学习了如何发送和接收SMS消息，然后学习了如何从Android应用程序中发送电子邮件。除了SMS消息和电子邮件，与外界通信的另一种方式是通过使用HTTP协议。使用HTTP协议，可以从一台Web服务器上下载数据。这方面一个很好的应用是与Web服务进行交互，这需要解析XML文件。

练　习

1. 说出在Android应用程序中可以用来发送SMS消息的两种方式。

2. 说出为发送和接收SMS消息需要在AndroidManifest.xml文件中声明的权限。

3. 如何通过BroadcastReceiver通知一个活动？

4. 说出为实现HTTP连接需要在AndroidManifest.xml文件中声明的权限。

练习答案参见附录C。

本章主要内容

主　　题	关 键 概 念
以编程方式发送SMS消息	使用SmsManager类
发送消息后获取反馈	在sendTextMessage()方法中使用两个PendingIntent对象
使用意图发送SMS消息	将意图类型设置为vnd.android-dir/mms-sms
接收SMS消息	实现一个BroadcastReceiver并在AndroidManifest.xml文件中设置它
使用意图发送电子邮件	将意图类型设置为message/rfc822
建立HTTP连接	使用HttpURLConnection类
访问Web服务	使用Document、DocumentBuilderFactory和DocumentBuilder类来解析由Web服务返回的XML结果

第9章 基于位置的服务

本章将介绍以下内容

- 如何在Android应用程序中显示Google Maps
- 如何在地图上显示缩放控件
- 如何在不同的地图视图间切换
- 如何在地图上添加标记
- 如何获取在地图上触摸的位置的地址
- 如何进行地理编码和反向地理编码
- 如何使用GPS、Cell-ID和Wi-Fi三角测量法来获取地理数据
- 如何监控一个位置

近些年来，我们都看到了移动应用程序的爆炸性增长。其中一类非常流行的应用程序是基于位置的服务，即LBS。LBS应用程序跟踪您的位置，并可提供额外服务，如定位附近的便利设施以及提供路线规划建议等。当然，LBS应用程序中的一个关键因素是地图，它可以对您的位置进行可视化表示。

本章中将学习如何在Android应用程序中使用Google Maps，以及如何以编程方式操作它。此外，还将学习如何利用Android SDK中提供的LocationManager类获得您的地理位置。

9.1 显示地图

Google Maps是与Android平台捆绑在一起的众多应用程序之一。除了直接使用Maps应用程序外，还可以将它嵌入到您自己的应用程序中来做一些非常酷的事情。本节介绍如何在Android应用程序中使用Google Maps以及用编程方式完成以下功能：

- 改变Google Maps的视图。
- 在Google Maps中获取位置的经度和纬度。
- 进行地理编码和反向地理编码(将一个地址转换为经纬度或反之)。
- 在Google Maps上添加标记。

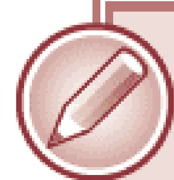
9.1.1 创建项目

开始时，需要首先创建一个Android项目以便在活动中显示Google Maps。

试一试 创建项目

LBS.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，按图9-1所示创建一个Android项目。



注意：为了在Android应用程序中使用Google Maps，需要确保选择Google APIs作为构建目标。Google Maps并不是标准Android SDK的一部分，因此需要在Google APIs附加组件中找到它。

(2) 创建项目后，可以看到在Google APIs文件夹下多出来一个JAR文件(maps.jar)，如图9-2所示。

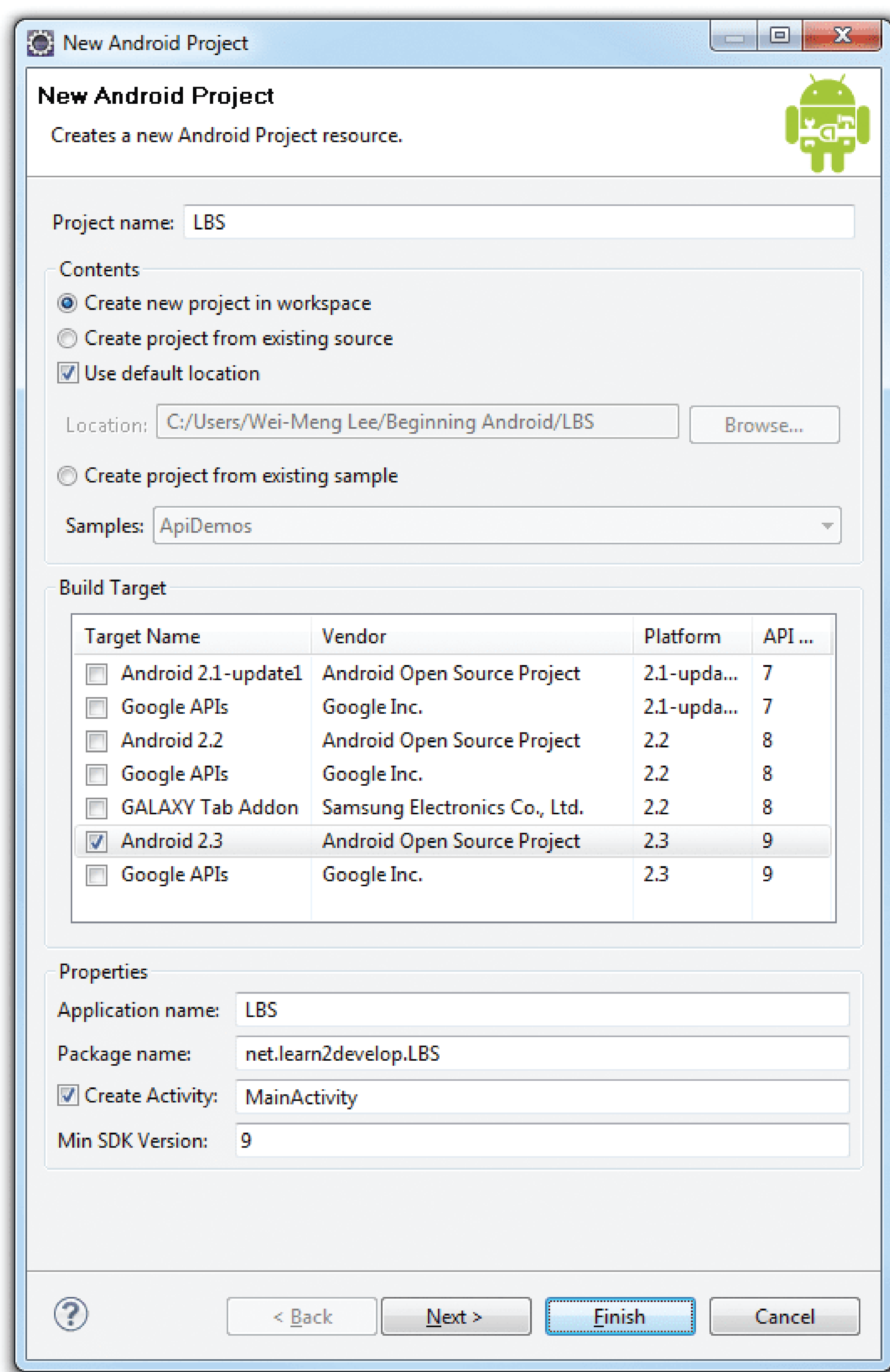


图 9-1

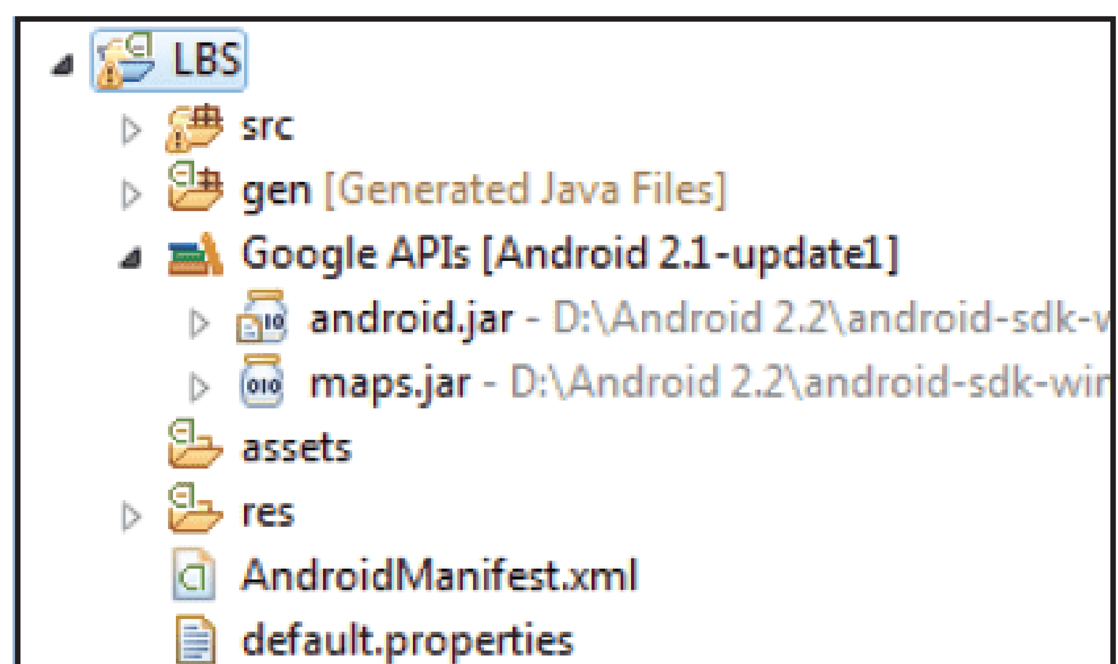


图 9-2

示例说明

这个简单的动作创建了一个使用Google APIs附加组件的Android项目。Google APIs附加组件包括了标准的Android库，以及打包在maps.jar文件中的Maps库。

9.1.2　获取Maps API密钥

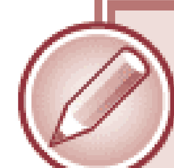
从Android SDK发行版1.0开始，在Android应用程序中集成Google Maps需要申请一个免费的Google Maps API密钥。当申请这个密钥时，必须同意Google的使用条款，所以一定要仔细阅读。

要申请一个密钥，可按照下面列出的一系列步骤进行。



注意：Google提供的关于申请Maps API密钥的详细文档参见http://code.google.com/android/add-ons/google-apis/mapkey.html。

首先，如果您正在直接连接到您的开发机的Android模拟器或Android设备上进行应用程序测试，则需要在默认文件夹(对于Windows 7的用户来说是C:\User\<username>\.android)下找到SDK调试证书。您可以进入Eclipse并选择Window | Preferences来验证调试证书是否存在。展开Android项，并选择Build(如图9-3所示)。在窗口的右侧，可以看到调试证书所在的位置。



注意：对于Windows XP用户来说，默认的Android文件夹是C:\Documents and Settings\<username>\Local Settings\Application Data\Android。

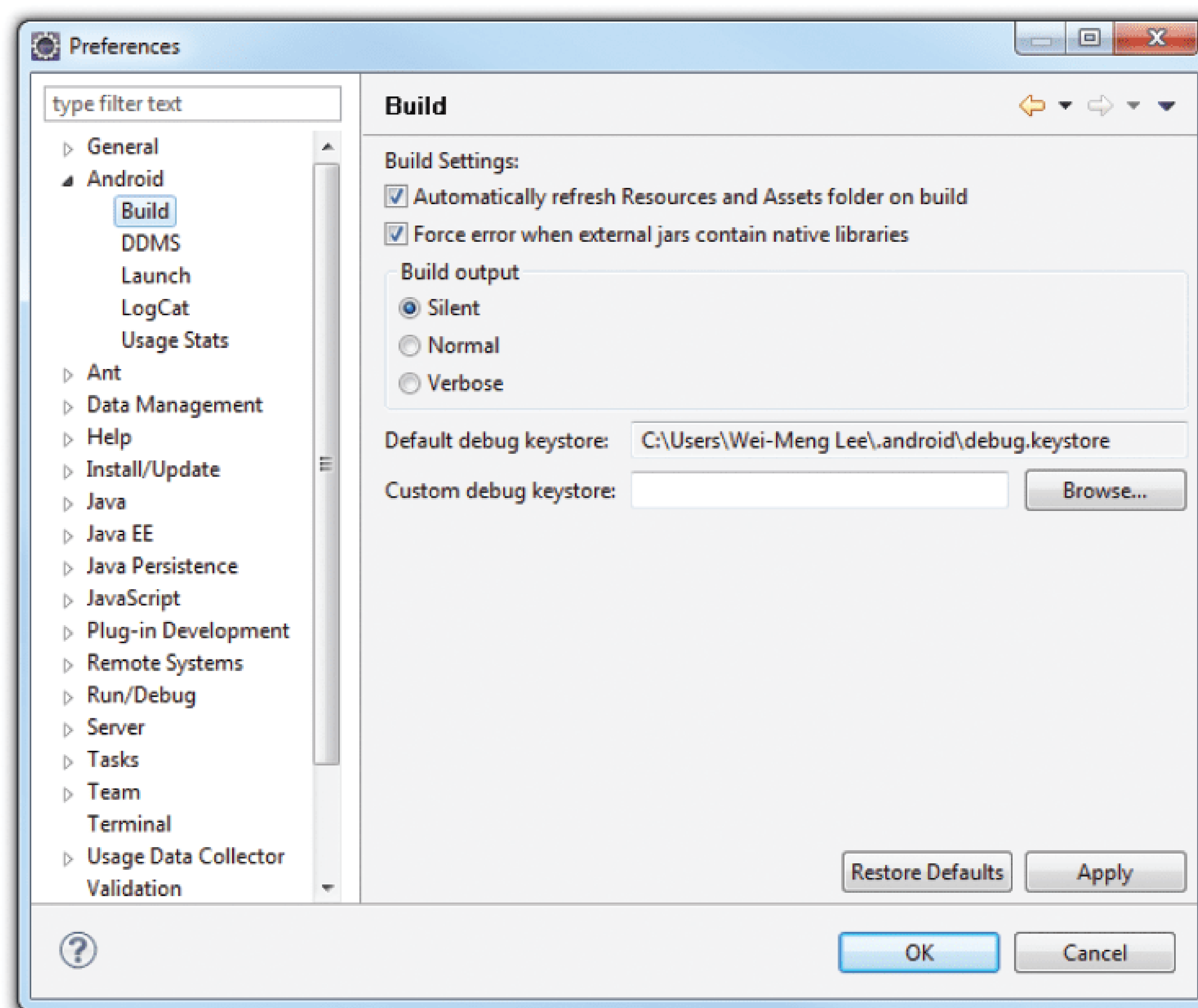


图 9-3

调试密钥库的文件名为debug.keystore。这是Eclipse用来为应用程序进行签名的证书，以便应用程序可以在Android模拟器或设备上运行。

使用调试密钥库，需要使用安装JDK时包含的keytool.exe应用程序来提取其MD5指纹。这个指纹是用来申请免费Google Maps密钥的。通常可以在C:\Program Files\Java\<JDK_version_number>\bin文件夹下找到keytool.exe。

发出以下命令(如图9-4所示)来提取MD5指纹：

```
keytool.exe -list -alias androiddebugkey -keystore
"C:\Users\<username>\.android\debug.keystore" -storepass android
-keypass android
```

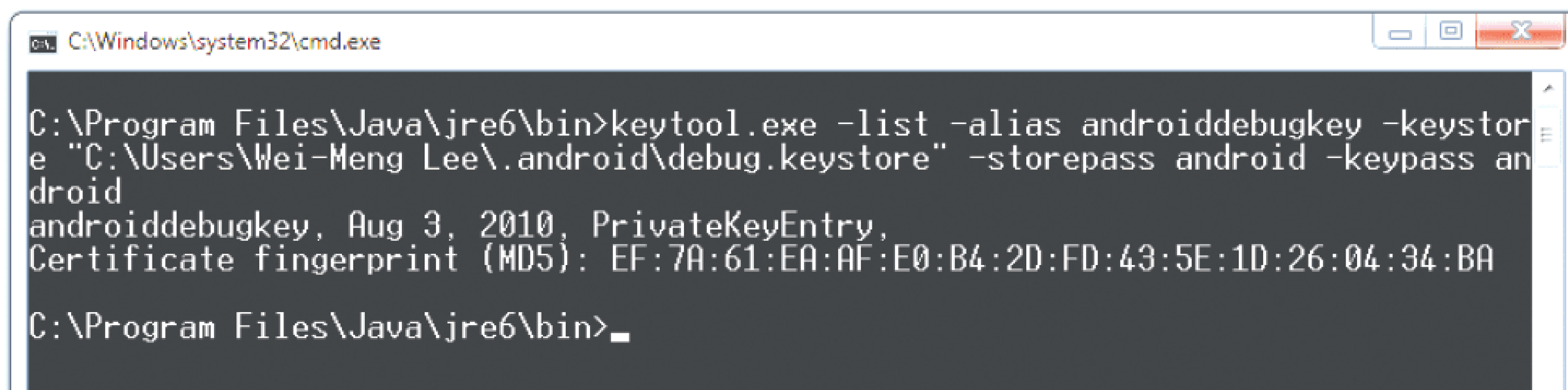


图 9-4

在这个例子中，我的MD5指纹是EF:7A:61:EA:AF:E0:B4:2D:FD:43:5E:1D:26:04:34:BA。

复制MD5证书指纹并将Web浏览器转到http://code.google.com/android/maps-api-signup.html。按照页面上的指令完成申请并获取Google Maps密钥。当这一切完成后，就应该可以看到如图9-5所示的类似信息。

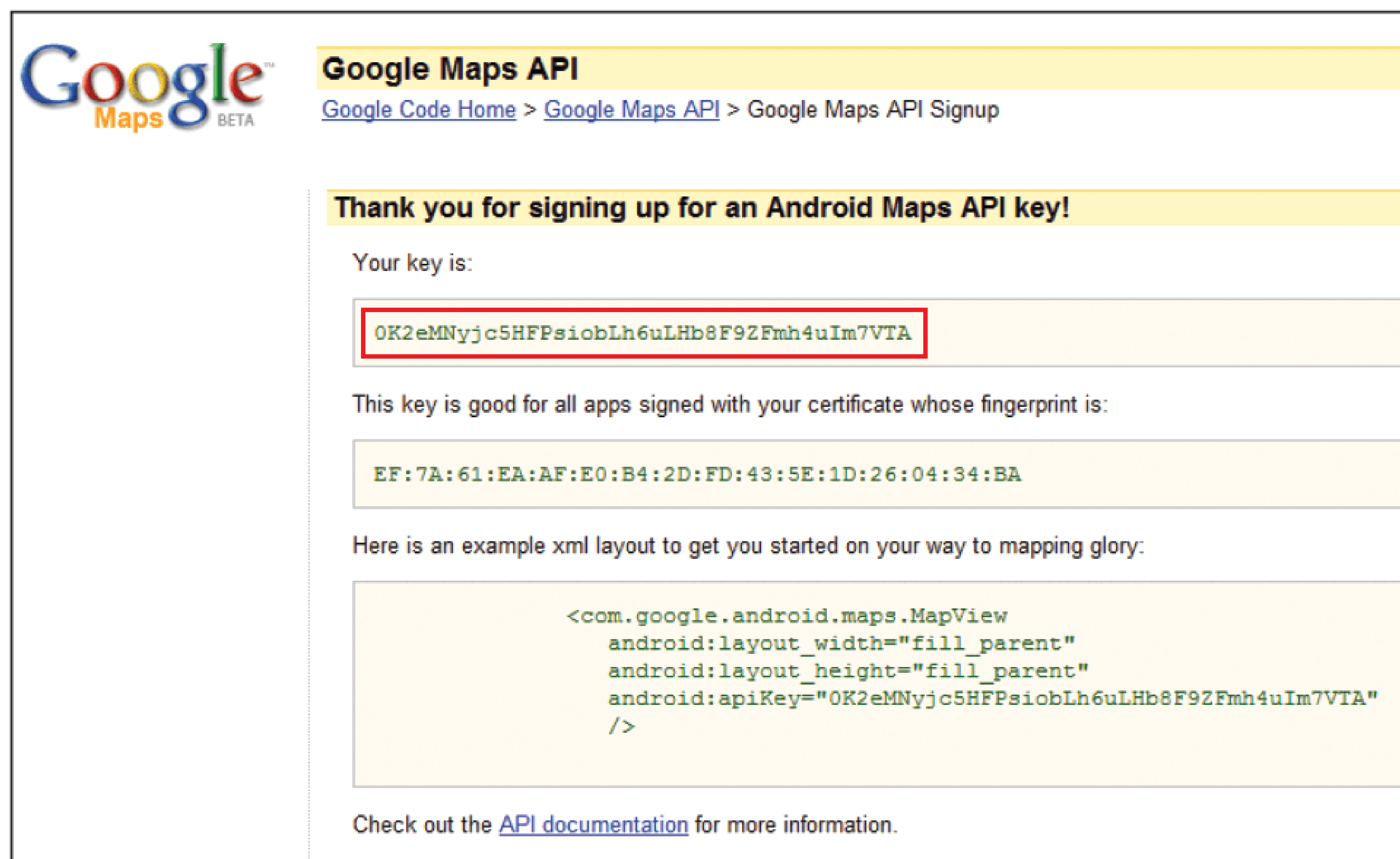
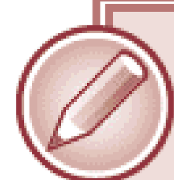


图 9-5



注意：虽然可以使用调试密钥库的MD5指纹来获取用于在Android模拟器或设备上调试应用程序的Maps API密钥，但如果试图将Android应用程序作为一个APK文件部署时，密钥将不再有效。一旦准备将应用程序部署到Android Market(或者使用其他发布方法)，就需要使用可以为您的应用程序进行签名的证书来重新申请一个Maps API密钥。第11章将对此进行详细讨论。

9.1.3　显示地图

现在就要准备在Android应用程序中显示Google Maps了。这将包含两个主要任务：

- 修改AndroidManifest.xml文件，添加<uses-library>元素和INTERNET权限。
- 在您的用户界面中添加MapView元素。

下面的“试一试”将告诉您应该如何做。

试一试　显示Google Maps

(1) 使用在前一节中所创建的项目，在main.xml文件中添加粗体显示的行(一定要用您先前获取的API密钥替换spiKey属性的值)：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<com.google.android.maps.MapView
    android:id="@+id/mapView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:enabled="true"
    android:clickable="true"
    android:apiKey="<YOUR KEY>" />
</LinearLayout>
```

(2) 在main.xml文件中添加下列粗体显示的行：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.LBS"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">

    <uses-library android:name="com.google.android.maps" />

        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="8" />
```

```
    <uses-permission android:name="android.permission.INTERNET"> </uses-permission>
</manifest>
```

(3) 在MainActivity.java文件中添加以下语句。注意，MainActivity现在扩展MapActivity类。

```
package net.learn2develop.LBS;

import android.app.Activity;
import android.os.Bundle;

import com.google.android.maps.MapActivity;

public class MainActivity extends MapActivity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    protected boolean isRouteDisplayed() {
        // TODO Auto-generated method stub
        return false;
    }
}
```

(4) 按F11键在Android模拟器上调试应用程序。图9-6展示了在应用程序的活动中显示的Google Maps。

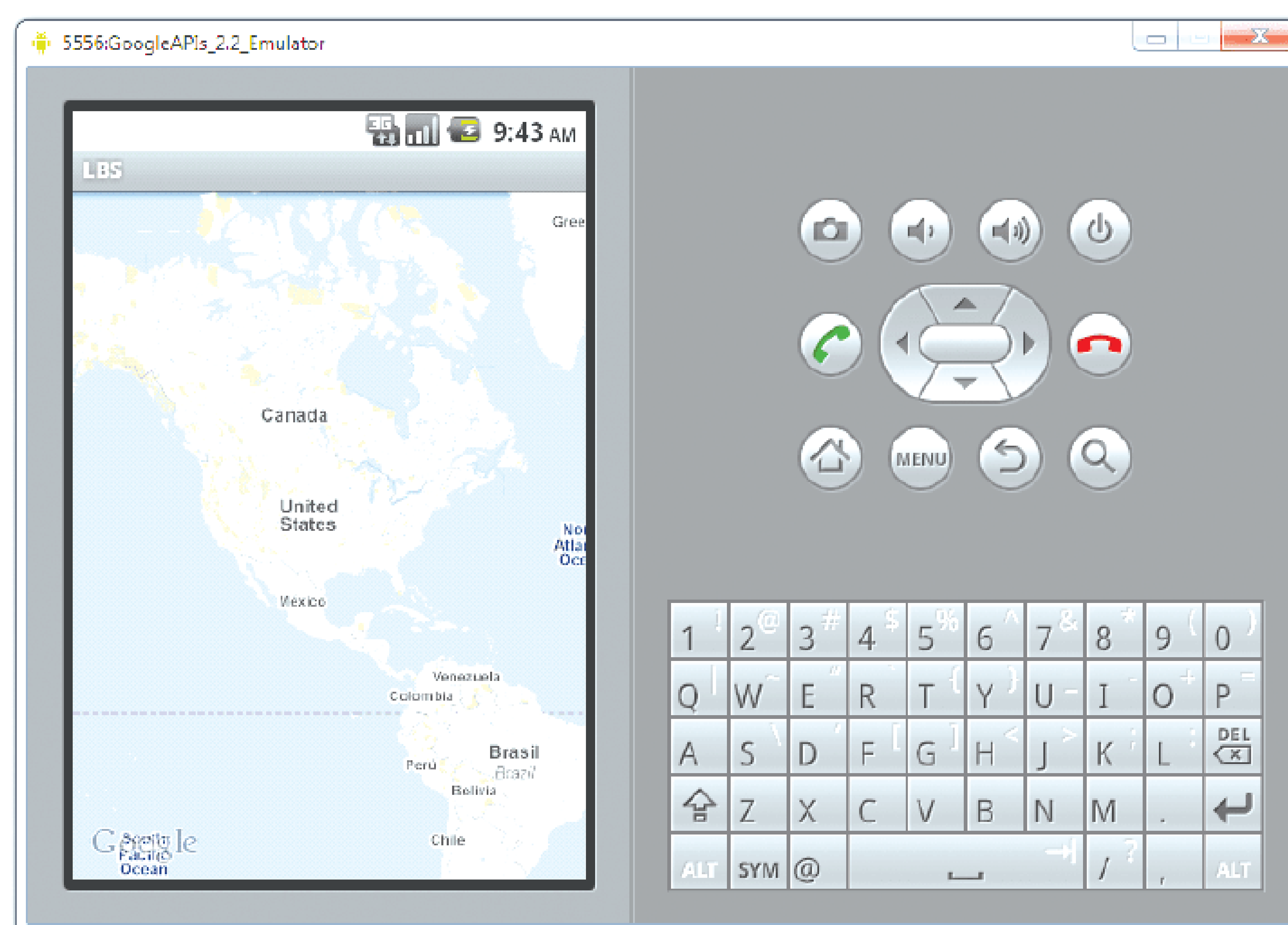


图 9-6

示例说明

为了在应用程序中显示Google Maps，首先需要在清单文件中有INTERNET权限。然后在UI文件中添加<com.google.android.maps.MapView>元素来把地图嵌入到活动中。特别重要的是，活动现在必须计算MapActivity类，而其本身是Activity类的扩展。对于MapActivity类，需要实现一个方法：isRouteDisplayed()。这一方法用于Google的计算目的，如果在地图上显示了路径信息，则此方法应返回true。对于一些最简单的情况，可以只是返回false。

如果看不到地图

如果您看到的只是一个带有网格的空白屏幕而没有显示Google Maps，那最有可能的是在main.xml文件中使用了错误的API密钥。还有可能是在AndroidManifest.xml文件中缺少INTERNET权限。最后，确保在您的模拟器/设备上具有Internet访问权限。

如果程序没有运行(也即程序崩溃)，那可能是您忘记在AndroidManifest.xml文件中添加以下语句：

```
<uses-library android:name="com.google.android.maps" />
```

注意这条语句在AndroidManifest.xml文件中的位置：它应该位于<Application>元素内。

9.1.4　显示缩放控件

9.1.3节展示了如何在Android应用程序中显示Google Maps。可以将地图平移到任何想要的位置上，地图能够随即进行动态更新。然而，在模拟器上是没有办法对地图上一个特定的位置直接进行放大或缩小的(在一个真正的Android设备上，可以用手指捏放地图进行缩放)。因此，在本节中将学习如何利用内置的缩放控件，使用户可以对地图进行放大或缩小的操作。

试一试　显示内置的缩放控件

(1) 打开在前面的活动中创建的项目，添加下列粗体显示的语句：

```
package net.learn2develop.LBS;

import android.app.Activity;
import android.os.Bundle;

import com.google.android.maps.MapActivity;

import com.google.android.maps.MapView;

public class MainActivity extends MapActivity {
    MapView mapView;
    /** 当活动第一次被创建时调用。 */
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mapView = (MapView) findViewById(R.id.mapView);
        mapView.setBuiltInZoomControls(true);
    }

    @Override
    protected boolean isRouteDisplayed() {
        // TODO Auto-generated method stub
        return false;
    }
}
```

(2) 按F11键在Android模拟器上调试应用程序。当单击并拖拽地图时，可以注意到在地图底部显示的内置缩放控件(如图9-7所示)。可以通过单击加(+)、减(−)图标来放大和缩小地图。

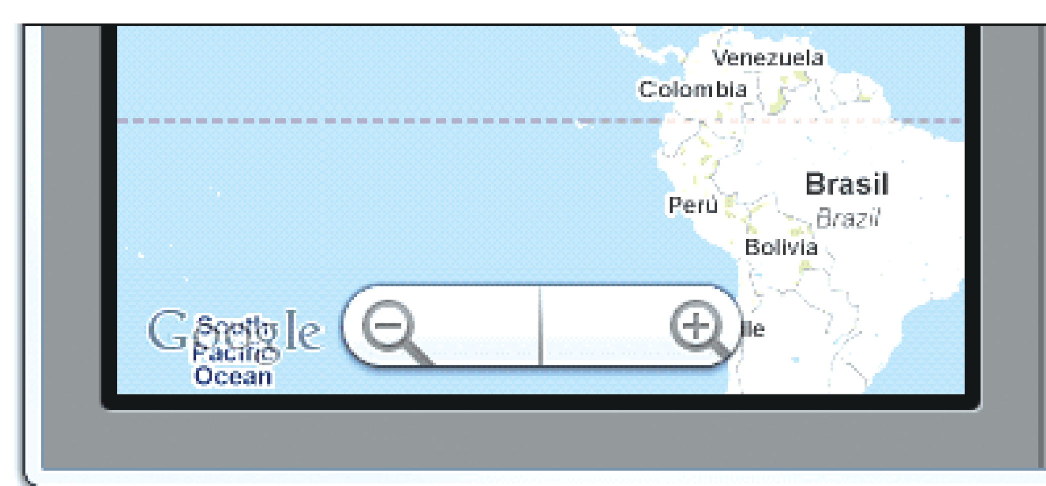


图 9-7

示例说明

要显示内置的缩放控件，首先要获取对地图的引用并调用setBuiltInZoomControls()方法：

```
mapView = (MapView) findViewById(R.id.mapView);
mapView.setBuiltInZoomControls(true);
```

除了显示缩放控件，还可以使用MapController类的zoomIn()或zoomOut()方法以编程方式来实现对地图的缩放。下面的“试一试”将告诉您如何做到这一点。

试一试 以编程方式缩放地图

(1) 使用在前面的活动中创建的项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.LBS;

import android.app.Activity;
import android.os.Bundle;

import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;

import android.view.KeyEvent;
import com.google.android.maps.MapController;

public class MainActivity extends MapActivity {
```

```
    MapView mapView;
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mapView = (MapView) findViewById(R.id.mapView);
        mapView.setBuiltInZoomControls(true);
    }

    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        MapController mc = mapView.getController();
        switch (keyCode)
        {
            case KeyEvent.KEYCODE_3:
                mc.zoomIn();
                break;
            case KeyEvent.KEYCODE_1:
                mc.zoomOut();
                break;
        }
        return super.onKeyDown(keyCode, event);
    }

    @Override
    protected boolean isRouteDisplayed() {
        // TODO Auto-generated method stub
        return false;
    }
}
```

(2) 按F11键在Android模拟器上调试应用程序。现在，可以通过按下模拟器上的数字键3来放大地图，按数字键1来缩小地图。

示例说明

为了处理在活动上的按键操作，需要处理onKeyDown事件：

```
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        //...
    }
```

为了管理对地图的平移和缩放，需要从MapView对象中获得一个MapController类的实例。MapController类包含了zoomIn()和zoomOut()方法(再加上其他一些用来控制地图的方法)，可以使用户对地图进行放大或缩小。

9.1.5 改变视图

默认情况下，Google Maps是以地图视图来显示的，它基本上描绘了感兴趣的街道和地方。还可以使用MapView类的setSatellite()方法将Google Maps设置为以卫星视图显示。

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mapView = (MapView) findViewById(R.id.mapView);
    mapView.setBuiltInZoomControls(true);
    mapView.setSatellite(true);
}
```

图9-8展示了以卫星视图显示的Google Maps。



图 9-8

除了卫星视图，也可以使用setStreetView()方法以街道视图显示地图(地图的所有街道被突出显示出来)。图9-9展示了分别以街道视图(左)和卫星视图(右)显示一个位置的地图。

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mapView = (MapView) findViewById(R.id.mapView);
    mapView.setBuiltInZoomControls(true);
    mapView.setSatellite(true);
    mapView.setStreetView(true);
}
```

如果想要在地图上显示交通状况，使用setTraffic()方法：

```
mapView.setTraffic(true);
```

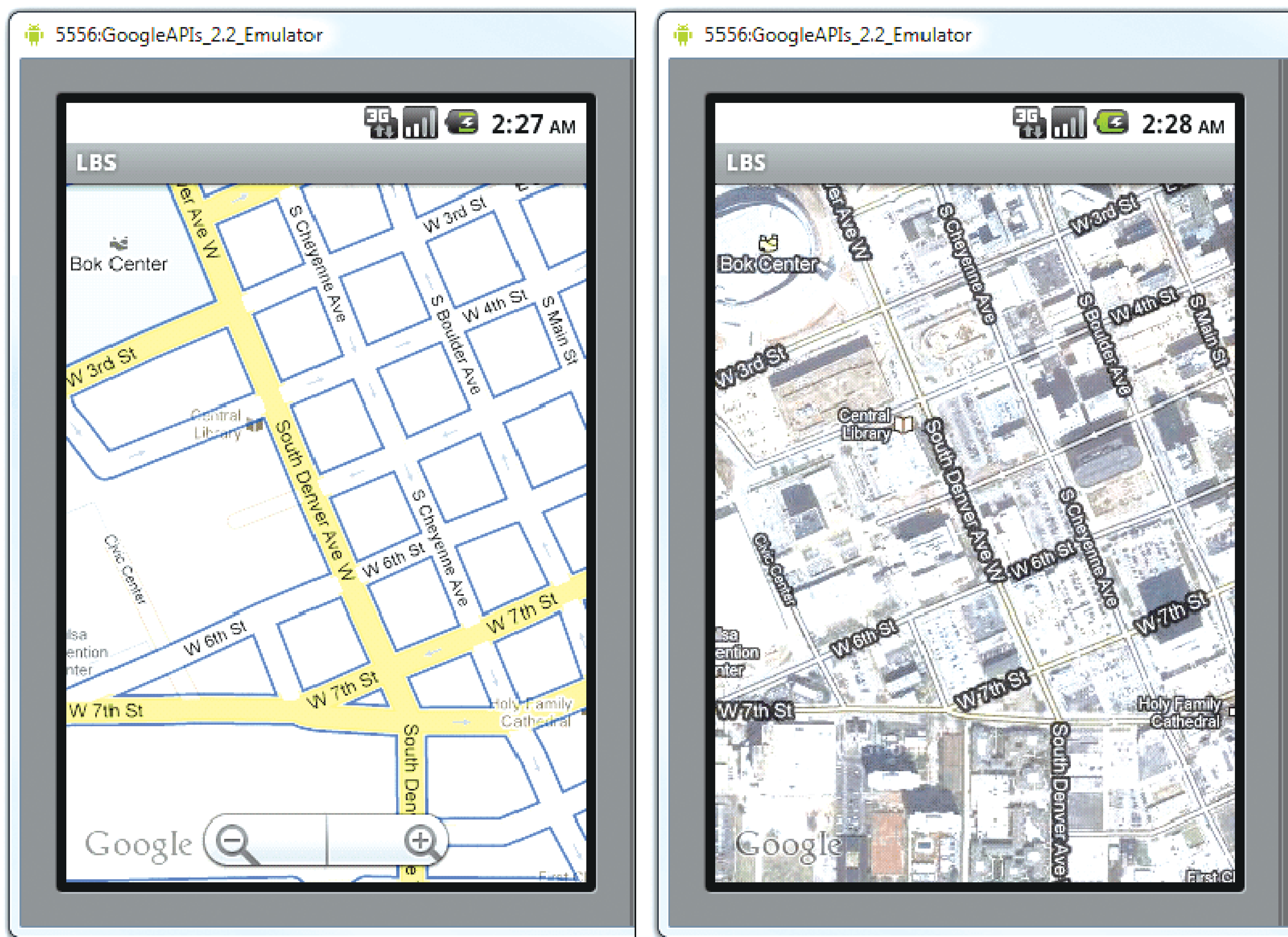


图 9-9

图9-10所示的地图上显示了当前的交通状况。不同的颜色反映不同的交通状况。一般来说，绿色相当于约每小时50英里的车速，交通比较顺畅；黄色相当于约每小时25～50英里的车速，交通状况中等；红色相当于低于约每小时25英里的车速，交通比较拥堵。

注意，随着新的国家和城市不断加入，仅仅美国、法国、英国、澳大利亚以及加拿大的主要城市提供交通状况信息。

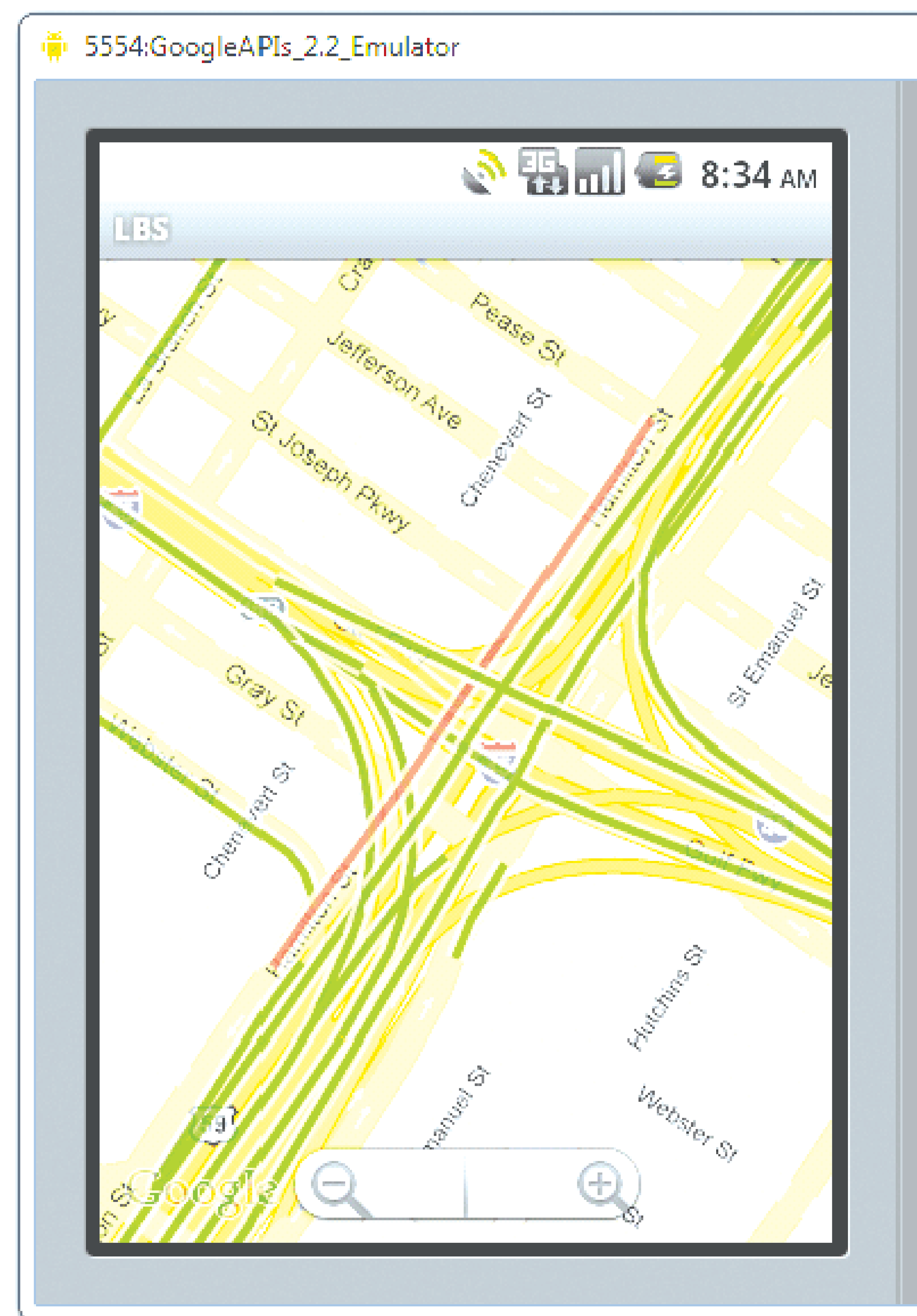


图 9-10

9.1.6　导航到特定位置

默认情况下，当Google Maps首次加载时，显示的是美国地图。然而，也可以将Google Maps设置为显示一个特定的位置。在这种情况下，可以使用MapController类的animateTo()方法。

下面的“试一试”向您展示了如何以编程方式将Google Maps动画显示到一个特定位置。

试一试　将地图导航到一个特定位置并进行显示

(1) 使用在前面的活动中创建的项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.LBS;

import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;

import com.google.android.maps.GeoPoint;

public class MainActivity extends MapActivity {
    MapView mapView;
    MapController mc;
    GeoPoint p;

    /** 当活动第一次被创建时调用。*/
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mapView = (MapView) findViewById(R.id.mapView);
        mapView.setBuiltInZoomControls(true);

        //mapView.setSatellite(true);

        mapView.setStreetView(true);


        mc = mapView.getController();
        String coordinates[] = {"1.352566007", "103.78921587"};
        double lat = Double.parseDouble(coordinates[0]);
        double lng = Double.parseDouble(coordinates[1]);

        p = new GeoPoint(
            (int) (lat * 1E6),
            (int) (lng * 1E6));

        mc.animateTo(p);
        mc.setZoom(13);
        mapView.invalidate();
    }

    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        MapController mc = mapView.getController();
        switch (keyCode)
        {
            case KeyEvent.KEYCODE_3:
```

```
                mc.zoomIn();
                break;
            case KeyEvent.KEYCODE_1:
                mc.zoomOut();
                break;
        }
        return super.onKeyDown(keyCode, event);
    }

    @Override
    protected boolean isRouteDisplayed() {
        // TODO Auto-generated method stub
        return false;
    }
}
```

(2) 按F11键在Android模拟器上调试应用程序。当地图被加载后，可注意到它动画显示到新加坡的一个特定位置(如图9-11所示)。

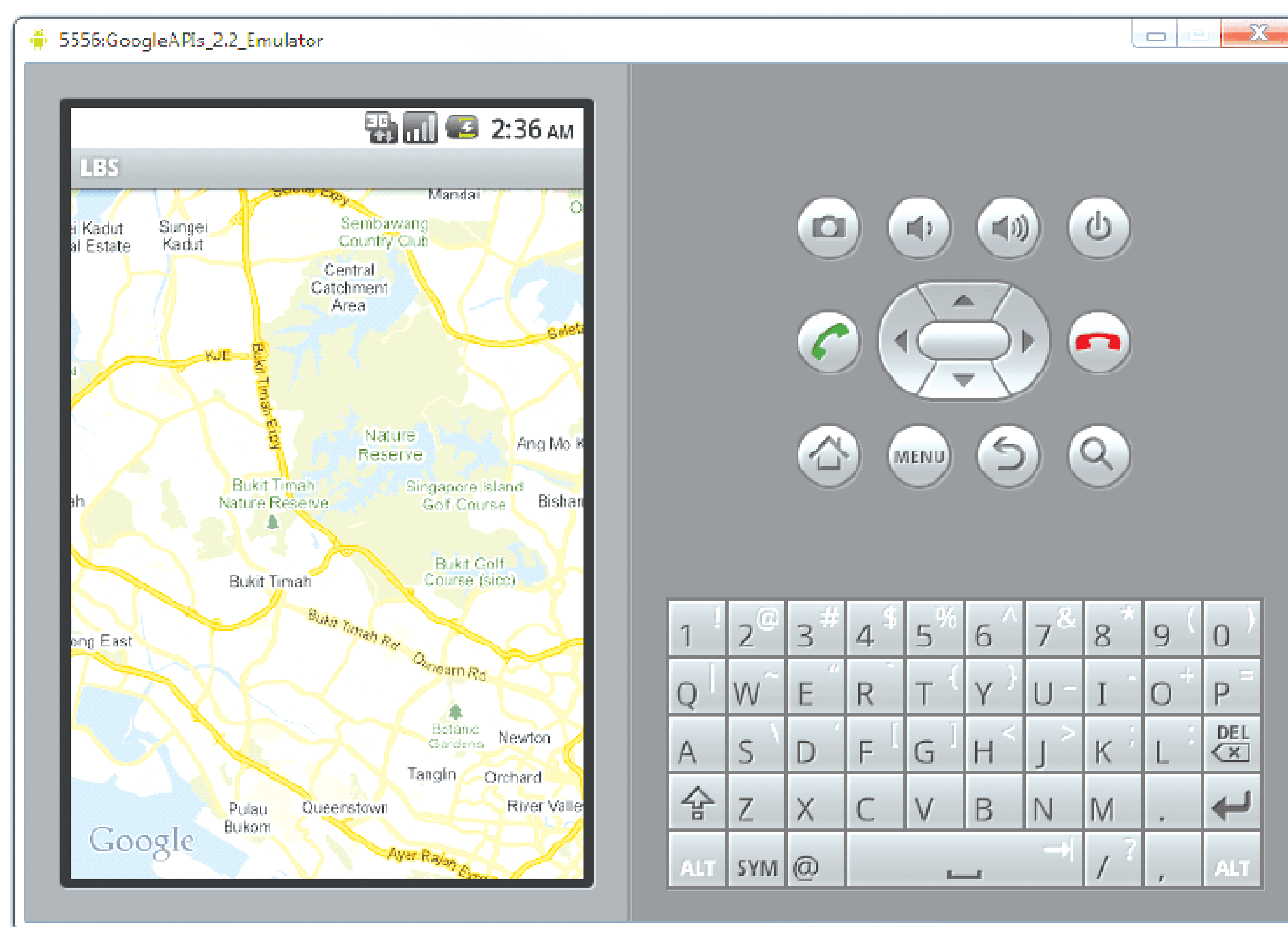


图 9-11

示例说明

在前面的代码中，首先从MapView实例中获取一个地图控制器并将其赋给一个MapController对象(mc)。然后使用一个GeoPoint对象来代表一个地理位置。注意，对于GeoPoint类，一个位置的经度和纬度是用微度来表示的。这意味着它们是以整数值存储。例如，对于一个纬度值40.747778，需要乘以1e6(10的6次方，一百万)得到40747778。

要将地图导航到特定位置，可以使用MapController类的animateTo()方法。setZoom()方法可以用来指定显示地图所采用的缩放级别(数字越大，地图上就能显示更多细节)。 invalidate()方法强制重绘MapView。

9.1.7 添加标记

在地图上添加标记来指明感兴趣的位置，这样可以使用户很容易地定位他们所要查找的地方。下面的“试一试”展示了如何在Google Maps上添加标记。

试一试 在地图上添加标记

(1) 创建一个包含一个图钉的GIF图像(如图9-12所示)，并将其复制到项目的res/drawable-mdpi文件夹下。为达到最好效果，将图像背景变为透明，以防止添加其后会遮挡住部分地图。

(2) 使用先前的活动中创建的项目，在MainActivity.java文件中添加下列粗体显示的语句：

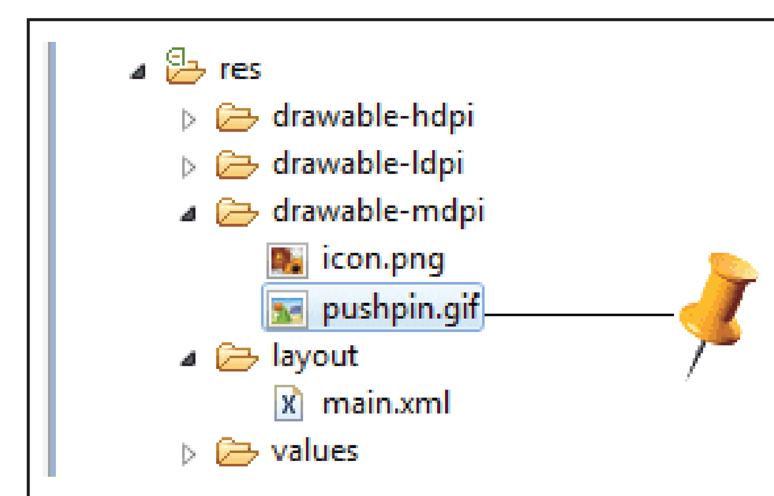


图 9-12

```
package net.learn2develop.LBS;

import android.app.Activity;

import android.os.Bundle;
import android.view.KeyEvent;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Point;
import com.google.android.maps.Overlay;
import java.util.List;

public class MainActivity extends MapActivity {
    MapView mapView;
    MapController mc;
    GeoPoint p;

    class MapOverlay extends com.google.android.maps.Overlay
    {
        @Override
        public boolean draw(Canvas canvas, MapView mapView,
        boolean shadow, long when)
        {
            super.draw(canvas, mapView, shadow);

            //---将GeoPoint转换为屏幕像素---
            Point screenPts = new Point();
            mapView.getProjection().toPixels(p, screenPts);

            //---添加标记---
```

```
            Bitmap bmp = BitmapFactory.decodeResource(
                getResources(), R.drawable.pushpin);
            canvas.drawBitmap(bmp, screenPts.x, screenPts.y-50, null);
            return true;
        }
    }

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mapView = (MapView) findViewById(R.id.mapView);
        mapView.setBuiltInZoomControls(true);

        //mapView.setSatellite(true);
        //mapView.setStreetView(true);

        mc = mapView.getController();
        String coordinates[] = {"1.352566007", "103.78921587"};
        double lat = Double.parseDouble(coordinates[0]);
        double lng = Double.parseDouble(coordinates[1]);

        p = new GeoPoint(
            (int) (lat * 1E6),
            (int) (lng * 1E6));

        mc.animateTo(p);
        mc.setZoom(13);

        //---添加一个位置标记---
        MapOverlay mapOverlay = new MapOverlay();
        List<Overlay> listOfOverlays = mapView.getOverlays();
        listOfOverlays.clear();
        listOfOverlays.add(mapOverlay);

        mapView.invalidate();
    }

    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        MapController mc = mapView.getController();
        switch (keyCode)
        {
            case KeyEvent.KEYCODE_3:
                mc.zoomIn();
                break;
            case KeyEvent.KEYCODE_1:
                mc.zoomOut();
```

```
                break;
            }
            return super.onKeyDown(keyCode, event);
        }
        @Override
        protected boolean isRouteDisplayed() {
            // TODO Auto-generated method stub
            return false;
        }
    }
```

(3) 按F11键在Android模拟器上调试应用程序。图9-13显示了添加到地图上的标记。



图 9-13

示例说明

为了在地图上添加标记，首先需要定义一个扩展Overlay类的类：

```
class MapOverlay extends com.google.android.maps.Overlay
{
    @Override
    public boolean draw(Canvas canvas, MapView mapView,
    boolean shadow, long when)
    {
        //...
    }
}
```

覆盖代表可以在地图上绘制的单独的一项。可以添加任意多个覆盖。在MapOverlay类中，重写draw()方法，这样就可以在地图上绘制出图钉图像。特别要注意的是，需要将地理位置(由GeoPoint对象p表示)转换成屏幕坐标：

```
//---将GeoPoint转换为屏幕像素---
  Point screenPts = new Point();
```

```
mapView.getProjection().toPixels(p, screenPts);
```

因为想让图钉的钉尖来指示地点的具体位置，所以需要从这个点(如图9-14所示)的y坐标扣除图像的高度(50像素)，并在该位置绘制图像：

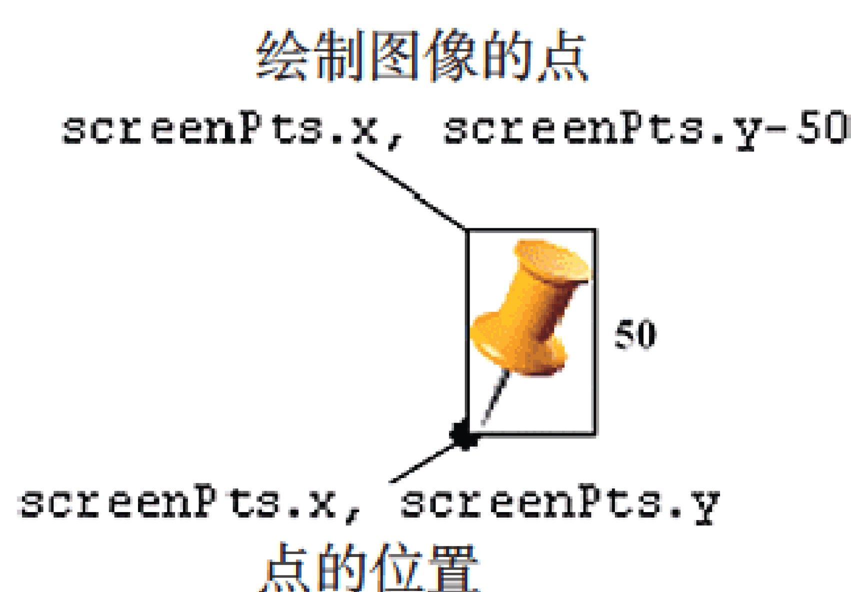


图 9-14

```
//---添加标记---
Bitmap bmp = BitmapFactory.decodeResource(
    getResources(), R.drawable.pushpin);
canvas.drawBitmap(bmp, screenPts.x, screenPts.y-50, null);
```

要添加标记，创建MapOverlay类的一个实例并把它添加到MapView对象的可用覆盖列表中：

```
//---添加一个位置标记---
MapOverlay mapOverlay = new MapOverlay();
List<Overlay> listOfOverlays = mapView.getOverlays();
listOfOverlays.clear();
listOfOverlays.add(mapOverlay);
```

9.1.8　获取触摸的位置

使用Google Maps一段时间之后，您可能想知道刚刚在屏幕上触碰到的位置对应地点的经纬度。知道这个信息非常有用，因为这样就可以确定一个位置的地址。这一过程被称为反向地理编码(我们将在9.1.9节学习如何做到这一点)。

如果已经在地图上添加了一个覆盖，可以在MapOverlay类中重写onTouchEvent()方法。每次用户触摸地图时都会触发这一方法。此方法有两个参数：MotionEvent和MapView。使用MotionEvent参数，可以利用getAction()方法来判断用户是否已经从屏幕上抬起了他/她的手指。在下面的代码片段中，如果用户触碰了屏幕，随即又抬起了手指，那么将显示出所触碰位置对应的经度和纬度。

```
import android.view.MotionEvent;
import android.widget.Toast;
//...

    class MapOverlay extends com.google.android.maps.Overlay
    {
        @Override
        public boolean draw(Canvas canvas, MapView mapView,
        boolean shadow, long when)
        {
            super.draw(canvas, mapView, shadow);
```

```
            //---将GeoPoint转换为屏幕像素---
            Point screenPts = new Point();
            mapView.getProjection().toPixels(p, screenPts);

            //---添加标记---
            Bitmap bmp = BitmapFactory.decodeResource(
                getResources(), R.drawable.pushpin);
            canvas.drawBitmap(bmp, screenPts.x, screenPts.y-50, null);
            return true;
        }

        @Override
        public boolean onTouchEvent(MotionEvent event, MapView mapView)
        {
            //---当用户抬起手指时---
            if (event.getAction() == 1) {
                GeoPoint p = mapView.getProjection().fromPixels(
                    (int) event.getX(),
                    (int) event.getY());
                    Toast.makeText(getBaseContext(),
                        "Location: "+
                        p.getLatitudeE6() / 1E6 + "," +
                        p.getLongitudeE6() /1E6 ,
                        Toast.LENGTH_SHORT).show();
            }
            return false;
        }
    }
```

getProjection()方法返回一个用于在屏幕像素坐标和经纬度坐标之间进行转换的投影。然后，fromPixels()方法将屏幕坐标转换成GeoPoint对象。

图9-15展示了当用户单击地图上一个位置时所显示的一组坐标。

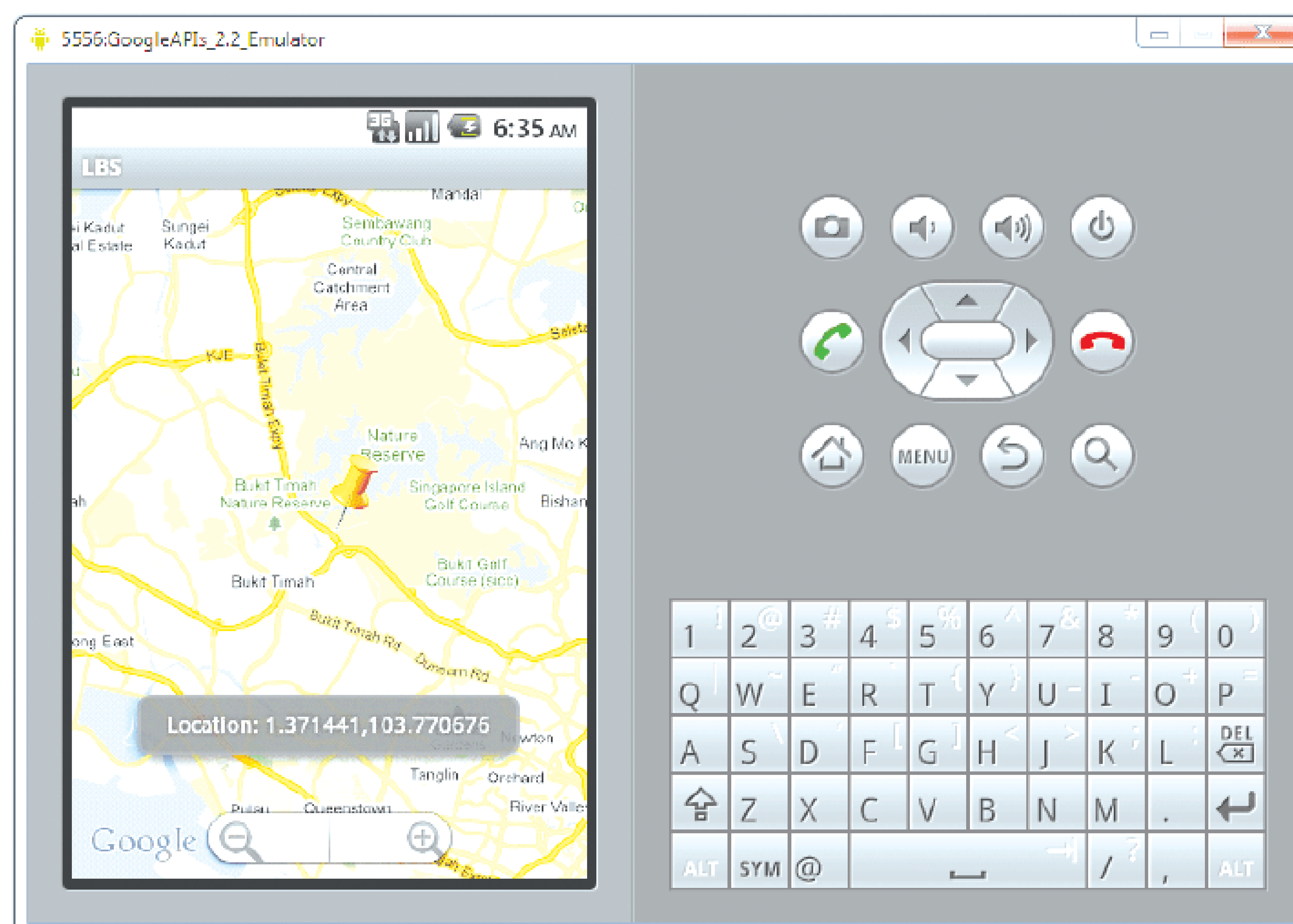


图 9-15

9.1.9　地理编码和反向地理编码

正如9.1.8节所述，如果知道某个位置的纬度和经度，就可以使用一个称为反向地理编码的过程来找到它的地址。Android中的Google Maps是通过Geocoder类来支持这一点的。下面的代码片段展示了如何利用getFromLocation()方法来获取您刚刚所触碰位置的地址：

```
import android.location.Address;
import android.location.Geocoder;
import java.util.Locale;
import java.io.IOException;
//...

        @Override
        public boolean onTouchEvent(MotionEvent event, MapView mapView)
        {
            //---当用户抬起手指时---
            if (event.getAction() == 1) {
                GeoPoint p = mapView.getProjection().fromPixels(
                    (int) event.getX(),
                    (int) event.getY());
                    /*
                    Toast.makeText(getBaseContext(),
                        "Location: "+
                        p.getLatitudeE6() / 1E6 + "," +
                        p.getLongitudeE6() /1E6 ,
                        Toast.LENGTH_SHORT).show();
                        */

                    Geocoder geoCoder = new Geocoder(
                        getBaseContext(), Locale.getDefault());
                    try {
                        List<Address> addresses = geoCoder.getFromLocation(
                            p.getLatitudeE6()  / 1E6,
                            p.getLongitudeE6() / 1E6, 1);

                        String add = "";
                        if (addresses.size() > 0)
                        {
                            for (int i=0; i<addresses.get(0).
                                  getMaxA ddressLineIndex(); i++)
                               add += addresses.get(0).getAddressLine(i) + "\n";
                        }
                        Toast.makeText(getBaseContext(), add, Toast.
                        LENGTH_SHORT).show();
                    }
                    catch (IOException e) {
                        e.printStackTrace();
                    }
                    return true;
            }
```

```
                return false;
            }
        }
```

Geocoder对象使用getFromLocation()方法将经度和纬度转换成一个地址。一旦获得地址之后，使用Toast类来显示它。图9-16展示了应用程序显示在地图上所触碰位置的地址。

如果知道一个位置的地址，但想要知道它的经度和纬度，那么可以通过地理编码做到这一点。同样，要达到此目的，可以使用Geocoder类。下面的代码演示了如何利用getFromLocationName()方法来获取帝国大厦的准确位置：

```
//---geo-coding---
Geocoder geoCoder = new Geocoder(this, Locale.getDefault());
try {
    List<Address> addresses = geoCoder.getFromLocationName(
        "empire state building", 5);

    String add = "";
    if (addresses.size() > 0) {
        p = new GeoPoint(
                (int) (addresses.get(0).getLatitude() * 1E6),
                (int) (addresses.get(0).getLongitude() * 1E6));
        mc.animateTo(p);
        mapView.invalidate();
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

图9-17显示了地图被导航到帝国大厦的位置。



图 9-16



图 9-17

9.2　获取位置数据

如今，移动设备普遍配备了GPS接收器。由于有许多卫星绕地球运行，因此使用GPS接收器可以很容易找到您所在的位置。然而，GPS工作时要求在空旷的地方，因此在室内或卫星无法穿透的地方(如穿山隧道)，GPS常常是无效的。

另一种用于定位的有效方式是通过发射塔三角测量法。当移动电话处于开机状态时，它会不断地与其周围的基站联系。在知晓发射塔的标识后，通过使用含有发射塔的标识和它们所处的确切地理位置的各种数据库，可以将这一信息转换为物理位置。发射塔三角测量法的优点是在室内也起作用，无须获得来自卫星的信息。然而，由于该方法的准确性取决于重叠信号的覆盖范围，其变化相当多，因此它不如GPS来得精确。发射塔三角测量法在人口稠密地区最为有效，因为那里的发射塔离得很近。

第三种定位方法是依靠Wi-Fi三角测量法。设备连接到Wi-Fi网络而不是连接到发射塔，并对照数据库来确定服务提供商所服务的位置。这里所描述的3种方法中，Wi-Fi三角测量法是最不准确的。

Android SDK提供了LocationManager类，可帮助您的设备确定用户的物理位置。下面的“试一试”展示了如何用代码做到这一点。

试一试　**使用LocationManager类将地图导航到特定位置**

(1) 使用前一节创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.LBS;

import android.app.Activity;
import android.content.Context;
//...
//...

import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;

public class MainActivity extends MapActivity {
    MapView mapView;
    MapController mc;
    GeoPoint p;

    private LocationManager lm;
    private LocationListener locationListener;

    class MapOverlay extends com.google.android.maps.Overlay
    {
        //...
    }

    /** 当活动第一次被创建时调用。 */
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mapView = (MapView) findViewById(R.id.mapView);
    mapView.setBuiltInZoomControls(true);

    mc = mapView.getController();

    //---首先导航到一个点---
    String coordinates[] = {"1.352566007", "103.78921587"};
    double lat = Double.parseDouble(coordinates[0]);
    double lng = Double.parseDouble(coordinates[1]);
    p = new GeoPoint(
        (int) (lat * 1E6),
        (int) (lng * 1E6));
    mc.animateTo(p);
    mc.setZoom(13);

    //---添加一个位置标记---
    //...
    //---反向地理编码---
    //...

    //---使用LocationManager类来获取位置数据---
    lm = (LocationManager)
        getSystemService(Context.LOCATION_SERVICE);

    locationListener = new MyLocationListener();

    lm.requestLocationUpdates(
            LocationManager.GPS_PROVIDER,
            0,
            0,
            locationListener);
}

private class MyLocationListener implements LocationListener
{
    @Override
    public void onLocationChanged(Location loc) {
        if (loc != null) {
            Toast.makeText(getBaseContext(),
                "Location changed : Lat: " + loc.getLatitude() +
                " Lng: " + loc.getLongitude(),
                Toast.LENGTH_SHORT).show();
        }

        p = new GeoPoint(
```

```
                    (int) (loc.getLatitude() * 1E6),
                    (int) (loc.getLongitude() * 1E6));

            mc.animateTo(p);
            mc.setZoom(18);
        }

        @Override
        public void onProviderDisabled(String provider) {
        }

        @Override
        public void onProviderEnabled(String provider) {
        }

        @Override
        public void onStatusChanged(String provider, int status,
            Bundle extras) {
        }
    }

    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        //...
    }

    @Override
    protected boolean isRouteDisplayed() {
        // TODO Auto-generated method stub
        return false;
    }
}
```

(2) 按F11键在Android模拟器上调试应用程序。

(3) 为了模拟Android模拟器收到的GPS数据，可以使用DDMS透视图中的Location Controls工具(如图9-18所示)。

(4) 首先确保已经在Devices选项卡中选中了模拟器，然后在Emulator Control选项卡中找到Location Controls工具，选择Manual选项卡。输入经度和纬度，然后单击Send按钮。

(5) 现在可观察到模拟器上的地图动画显示到另一个位置(如图9-19所示)。这证明应用程序已经收到了GPS数据。

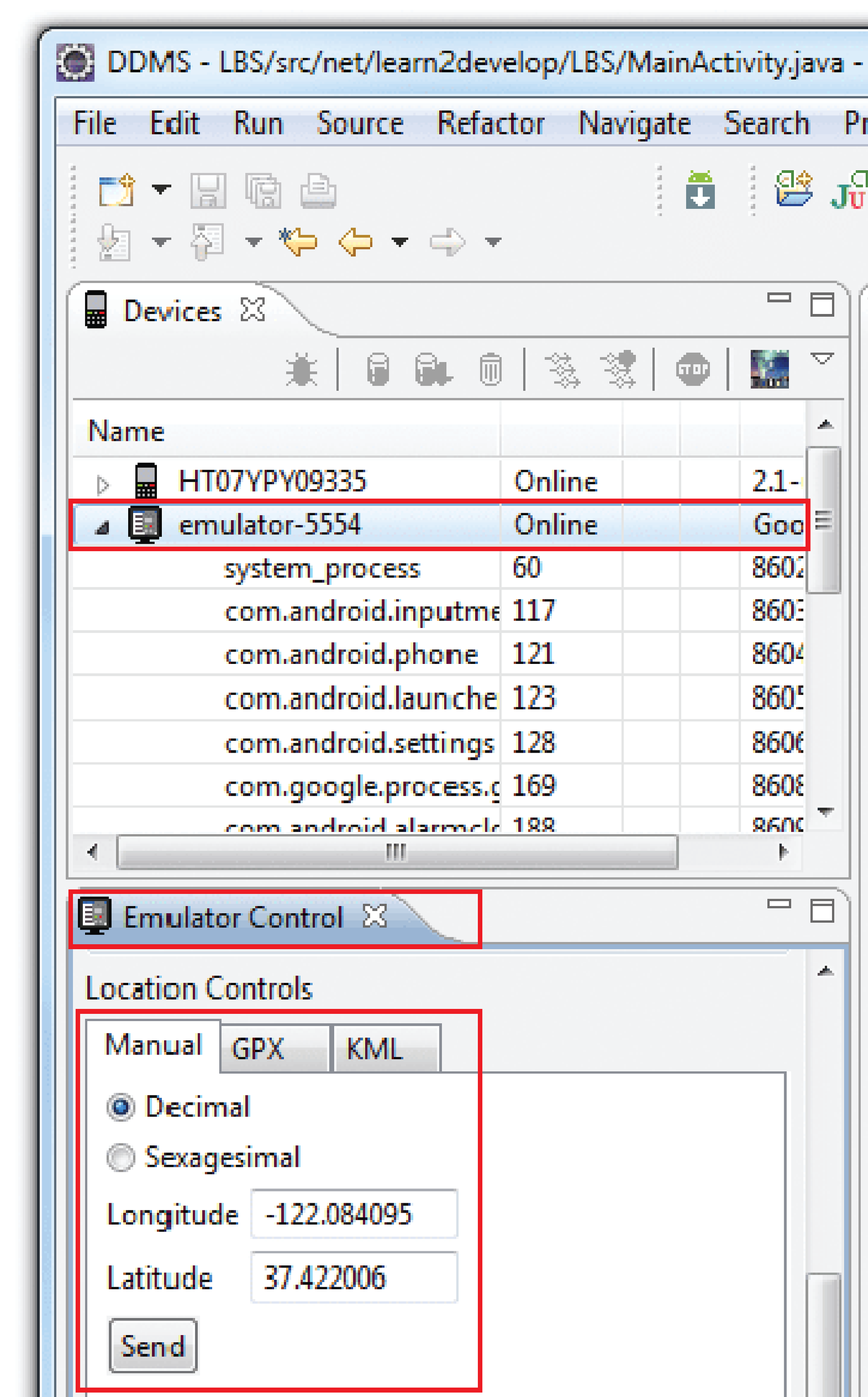


图 9-18

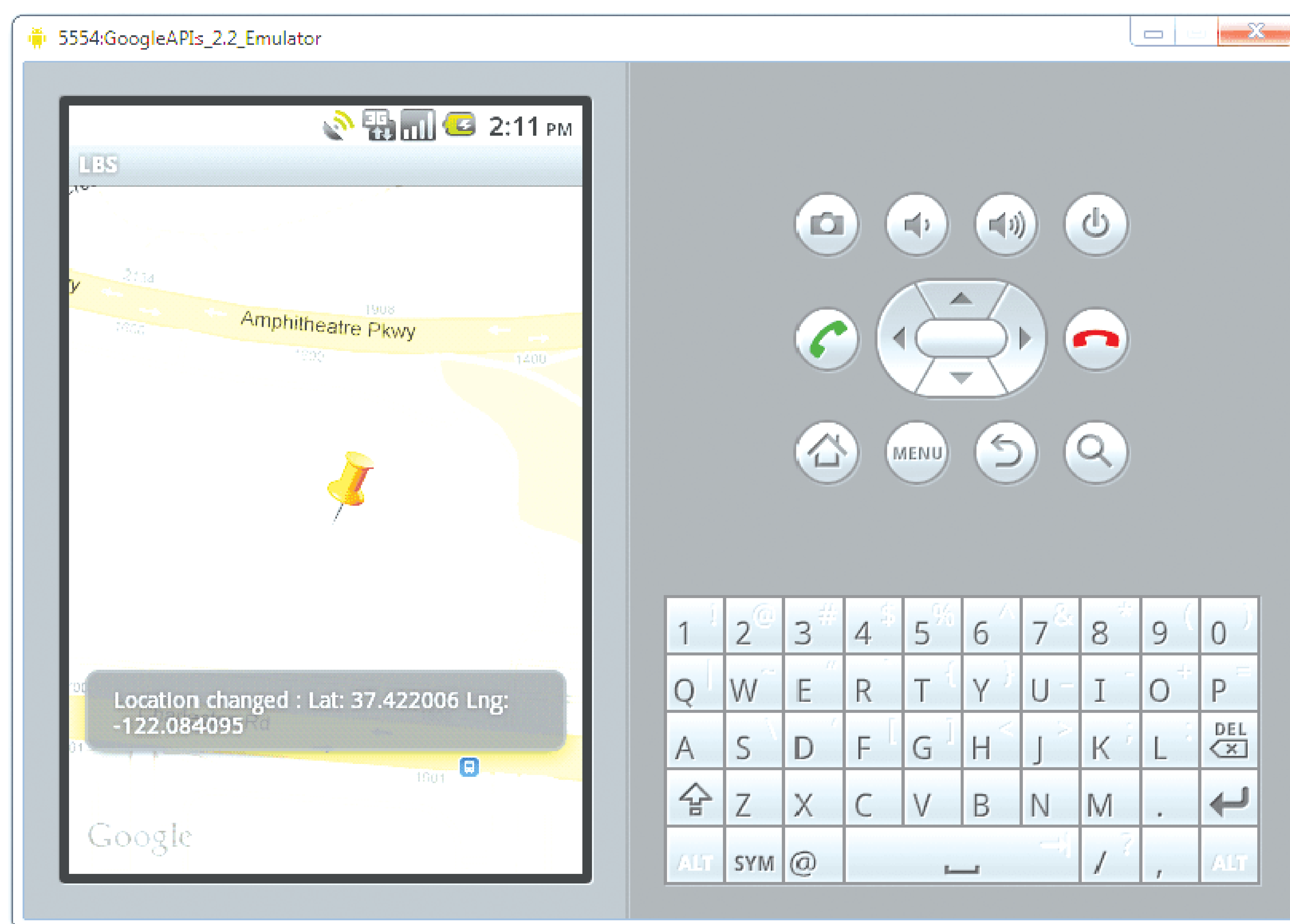


图 9-19

示例说明

在Android中，基于位置的服务是由LocationManager类提供的，位于android.location包中。使用LocationManager类，应用程序可以定期获取设备的地理位置的更新，并在它进入某个位置附近时触发一个意图。

在MainActivity.java文件中，首先使用getSystemService()方法获取一个指向LocationManager类的引用。为了在位置有变化时得到通知，需要注册一个位置变化的请求，这样才会定期地通知您的程序。这是通过requestLocationUpdates()方法来完成的：

```
lm.requestLocationUpdates(
        LocationManager.GPS_PROVIDER,
        0,
        0,
        locationListener);
```

这个方法接受4个参数：

- provider——您所注册的服务提供商的名称。在本例中，使用GPS来获取地理位置数据。
- minTime——进行通知的最短时间间隔，用毫秒作为单位。
- minDistance——进行通知的最短距离，用米作为单位。
- listener——一个对象，在每一次位置更新时将调用其onLocationChanged()方法。

MyLocationListener类实现了LocationListener抽象类。在该实现中需要重写以下4个方法：

- onLocationChanged(Location location)——当位置改变时调用
- onProviderDisabled(String provider)——当提供商被用户禁用时调用
- onProviderEnabled(String provider)——当提供商被用户启用时调用
- onStatusChanged(String provider, int status, Bundle extras)——当提供商状态改变时调用

在这个例子中，我们对于一个位置变化时到底发生了什么更感兴趣。因此，可以在onLocation-

Changed()方法中写一些代码来看一下。具体来说，当一个位置发生变化时，在屏幕上会显示一个小的对话框来表明新的位置信息：经度和纬度。可以使用Toast类显示此对话框。

如果想使用Cell-ID和Wi-Fi三角测量法(对于室内使用很重要)获得您的位置数据，那么可以使用网络位置服务提供商，如下所示：

```
lm.requestLocationUpdates(
        LocationManager.NETWORK_PROVIDER,
        0,
        0,
        locationListener);
```

在应用程序中，可以把GPS位置服务提供商和网络位置服务提供商结合起来。

监控一个位置

LocationManager类的一个非常酷的功能是它能够监视一个特定的位置。这是使用addProximityAlert()方法来实现的。下面的代码片段显示了如何监控一个特定位置。如果用户位于从该位置起5米的半径内，应用程序将触发一个意图来启动Web浏览器：

```
//---使用LocationManager类来获取位置数据---
lm = (LocationManager)
    getSystemService(Context.LOCATION_SERVICE);

//---如果用户正处于某个位置，使用PendingIntent来调用活动---
PendingIntent pendIntent = PendingIntent.getActivity(
    this, 0, new
    Intent(android.content.Intent.ACTION_VIEW,
      Uri.parse("http://www.amazon.com")), 0);

lm.addProximityAlert(37.422006, -122.084095, 5, -1, pendIntent);
```

addProximityAlert()方法接受5个参数：纬度、经度、半径(以米为单位)、有效期限(接近警报的有效时间，时间过后将删除警报；−1表示没有有效期限)，以及挂起的意图。

注意，如果Android设备的屏幕进入睡眠状态，也是每4分钟检查一次接近状态，这样可以以延长设备的电池寿命。

9.3　本章小结

本章对在Android应用程序中用于显示Google Maps的MapView对象进行了大致的了解。我们学习了操作地图的不同方式，还学习了如何使用不同的网络服务提供商来获取地理位置数据：采用GPS、Cell-ID和Wi-Fi三角测量法。

练　习

1. 如果您在Android应用程序中嵌入了Google Maps API，但在应用程序加载时并没有显示出地图，那么最可能的原因有哪些?
2. 地理编码和反向地理编码有何区别?
3. 说出可用于获得位置数据的两个位置服务提供商。
4. 监控一个位置的方法是什么?

练习答案参见附录C。

本章主要内容

主　　题	关 键 概 念
显示MapView	<com.google.android.maps.MapView android:id="@+id/mapView" android:layout_width="fill_parent" android:layout_height="fill_parent" android:enabled="true" android:clickable="true" android:apiKey="0K2eMNyjc5HFPsiobLh6uLHb8F9ZFmh4uIm7VTA" />
引用Map库	<uses-library android:name="com.google.android.maps" />
显示缩放控件	mapView.setBuiltInZoomControls(true);
以编程方式对地图进行缩放	mc.zoomIn(); mc.zoomOut();

改变视图	mapView.setSatellite(true);
	mapView.setStreetView(true);
	mapView.setTraffic(true);

动画显示到一个特定位置	mc = mapView.getController();
	String coordinates[] = {"1.352566007", "103.78921587"};
	double lat = Double.*parseDouble*(coordinates[0]);
	double lng = Double.*parseDouble*(coordinates[1]);
	p = new GeoPoint(
	(int) (lat * 1E6),
	(int) (lng * 1E6));
	mc.animateTo(p);

添加标记	实现一个Overlay类并重写draw()方法
获取在地图上触摸的位置	GeoPoint p = mapView.getProjection().fromPixels((int) event.getX(), (int) event.getY());

(续表)

主　题	关键概念
地理编码和反向地理编码	使用Geocoder类

获取位置数据

```
private LocationManager lm;

//...

        lm = (LocationManager)
            getSystemService(Context.LOCATION_SERVICE);

        locationListener = new MyLocationListener();

        lm.requestLocationUpdates(
                LocationManager.GPS_PROVIDER,
                0,
                0,
                locationListener);

//...

    private class MyLocationListener implements LocationListener
    {
        @Override
        public void onLocationChanged(Location loc) {
            if (loc != null) {
            }
        }

        @Override
        public void onProviderDisabled(String provider) {
        }

        @Override
        public void onProviderEnabled(String provider) {
        }

        @Override
        public void onStatusChanged(String provider,
            int status, Bundle extras) {
        }
    }
```

监控一个位置

```
lm.addProximityAlert(37.422006, -122.084095, 5, -1, pendIntent);
```

第10章 开发Android服务

本章将介绍以下内容

- 如何创建一个在后台运行的服务
- 如何在一个单独的线程中执行长时间运行的任务
- 如何在一个服务中执行重复的任务
- 如何在活动和服务间进行通信

服务是Android中一个在后台运行的应用程序，它无须与用户进行交互。例如，在使用一个应用程序时，您可能想要在同一时间播放背景音乐。在这种情况下，播放背景音乐的代码就不需要与用户进行交互，因此它可以作为一个服务来运行。对于不需要向用户展示用户界面的情况下，使用服务也是一个理想的选择。用来持续记录设备所在地理坐标的应用程序就是一个很好的示例。在这种情况下，可以写一个服务在后台执行任务。在本章中，将介绍如何创建自己的服务以及利用它们来异步地执行后台任务。

10.1 创建自己的服务

理解服务如何工作的最好方法就是亲自创建一个服务。下面的“试一试”向您展示了创建一个简单服务的步骤。后续章节的内容将为这个服务增加更多的功能。至于现在，先学习如何启动和停止服务。

试一试 创建一个简单服务

Services.zip代码文件可以在Wrox.com上下载

(1) 打开Eclipse，按图10-1所示创建一个新的Android项目。

(2) 在项目中添加一个新的名为MyService.java的类文件，其中的代码如下所示：

```
package net.learn2develop.Services;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
```

```
import android.widget.Toast;

public class MyService extends Service {

    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // 我们希望这个服务在被显式停止前一直运行，所以返回“粘性的”状态。
        Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
        return START_STICKY;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Toast.makeText(this, "Service Destroyed", Toast.LENGTH_LONG).show();
    }
}
```

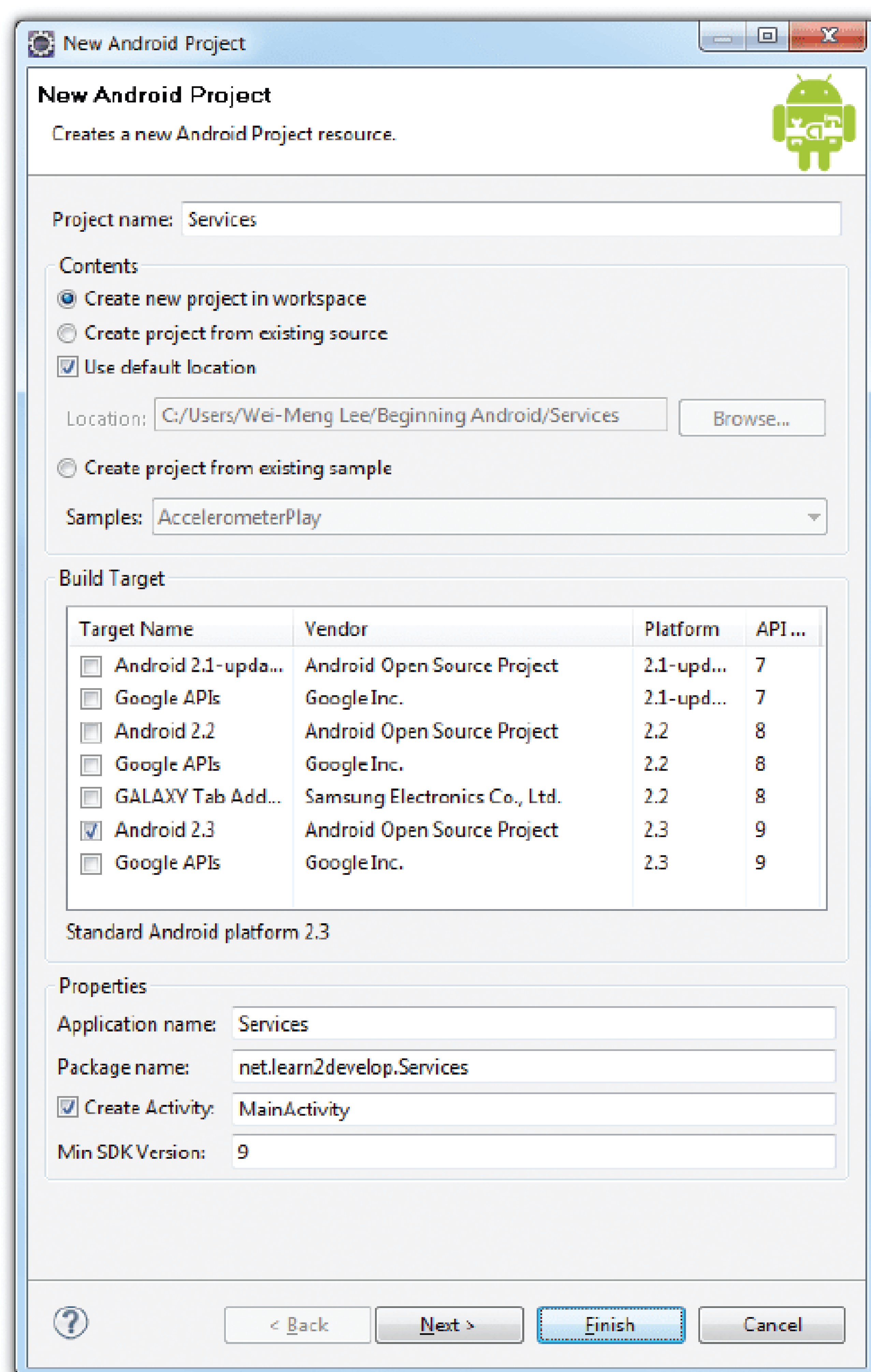


图 10-1

(3) 在AndroidManifest.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Services"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".MyService" />
    </application>
    <uses-sdk android:minSdkVersion="9" />
</manifest>
```

(4) 在main.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<Button android:id="@+id/btnStartService"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Start Service" />
<Button android:id="@+id/btnStopService"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Stop Service" />
</LinearLayout>
```

(5) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Services;

import android.app.Activity;
import android.os.Bundle;

import android.content.Intent;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
```

```
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);

            Button btnStart = (Button) findViewById(R.id.btnStartService);
            btnStart.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                    startService(new Intent(getBaseContext(), MyService.class));
                }
            });

            Button btnStop = (Button) findViewById(R.id.btnStopService);
            btnStop.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                    stopService(new Intent(getBaseContext(), MyService.class));
                }
            });
        }
    }
```

(6) 按F11键在Android模拟器上调试应用程序。

(7) 单击Start Service按钮将启动服务(如图10-2所示)。要停止服务，可单击Stop Service按钮。

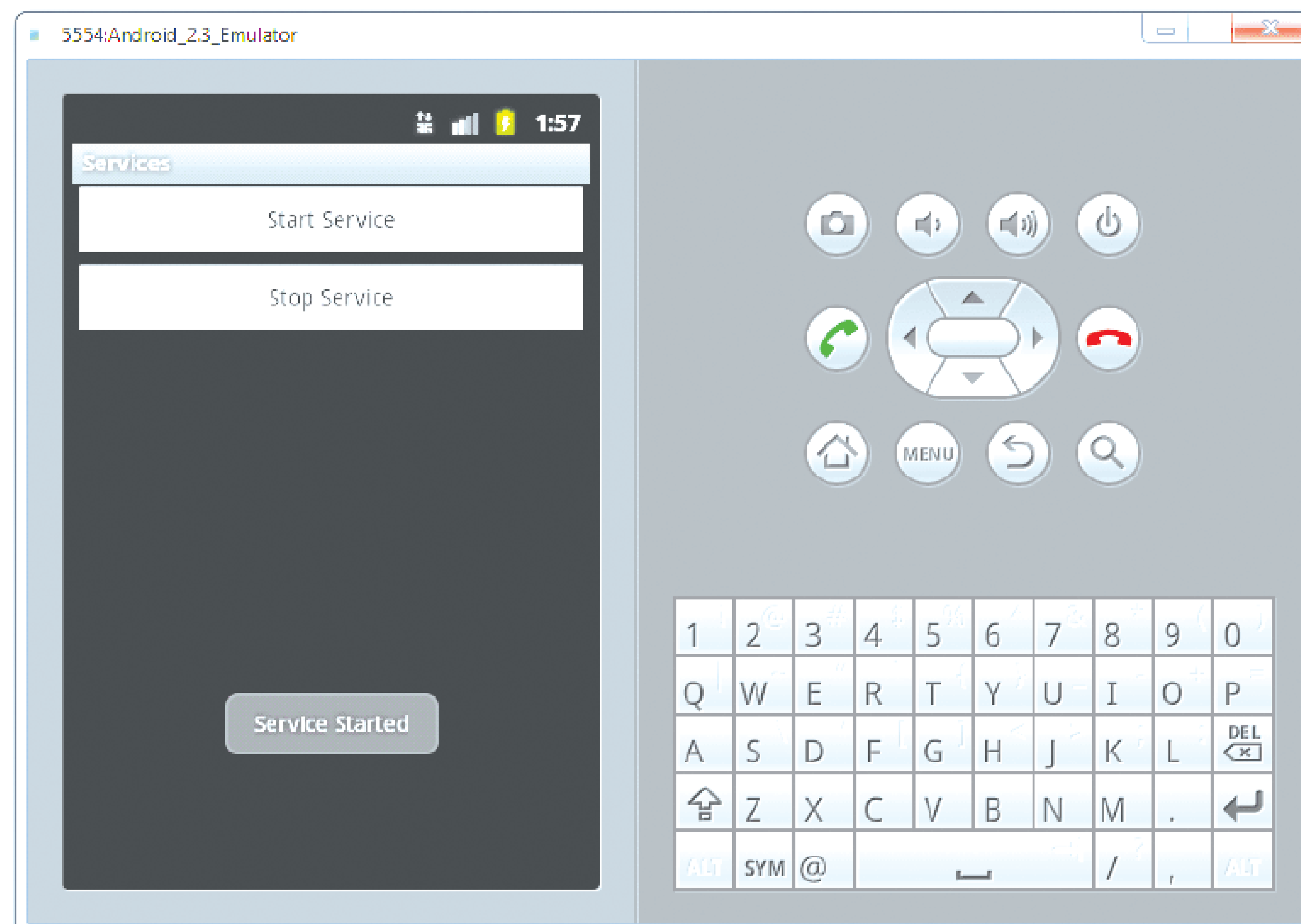


图 10-2

示例说明

这个示例展示了您可以创建的最简单的服务。当然，服务本身是没有做什么有用的事情，但它足以说明整个创建的过程。

首先，它定义了一个扩展Service基类的类。所有服务都扩展Service类：

```
public class MyService extends Service {

}
```

在MyService类中，实现了3个方法：

```
@Override
public IBinder onBind(Intent arg0) {...}

@Override
public int onStartCommand(Intent intent, int flags, int startId) {...}

@Override
public void onDestroy() {...}
```

onBind()方法可以用来将一个活动绑定到一个服务。这反过来又使活动可以直接访问服务内部的成员和方法。目前，只是为这个方法返回null。在本章的后面，您将学习更多有关绑定的内容。

当使用startService()方法(稍后讨论)显式启动服务时将调用onStartCommand()方法。这个方法意味着服务的启动，并为服务做一些需要做的事情。此方法返回常量START_STICKY，因此只要服务不被显式地停止，它都将继续运行。

当使用stopService()方法停止服务时将调用onDestroy()方法。这一方法将清除服务所使用的资源。

所有已创建的服务必须在AndroidManifest.xml文件中声明，如下所示：

```
<service android:name=".MyService" />
```

如果想让其他应用程序也可使用您的服务，可以添加一个带有动作名称的意图筛选器，如下所示：

```
<service android:name=".MyService">
    <intent-filter>
        <action android:name="net.learn2develop.MyService" />
    </intent-filter>
</service>
```

要启动一个服务，使用startService()方法，如下所示：

```
startService(new Intent(getBaseContext(), MyService.class));
```

如果调用一个外部服务，要按如下所示调用startService()方法：

```
startService(new Intent("net.learn2develop.MyService"));
```

要停止一个服务，使用stopService()方法，如下所示：

```
stopService(new Intent(getBaseContext(), MyService.class));
```

10.1.1　在服务中执行长时间运行的任务

由于在上一节中所创建的服务没有做任何有用的事情，因此在本节中将修改这一服务，使其可以执行一个任务。在下面的“试一试”中，将模拟一个从Internet上下载文件的服务。

试一试　使服务变得有用

(1) 使用在前一节中所创建的同一个项目，在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Services;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;

import java.net.MalformedURLException;
import java.net.URL;

public class MyService extends Service {
    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        //我们希望这个服务在被显式停止前一直运行，所以返回“粘性的”状态。
        Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();

        try {
            int result = DownloadFile(new URL("http://www.amazon.com/somefile.pdf"));
            Toast.makeText(getBaseContext(),
                "Downloaded " + result + " bytes",
                Toast.LENGTH_LONG).show();
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return START_STICKY;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Toast.makeText(this, "Service Destroyed", Toast.LENGTH_LONG).show();
    }
```

```
        private int DownloadFile(URL url) {
            try {
                //---模拟花一些时间来下载文件---
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                 e.printStackTrace();
            }
            //---返回一个表示下载文件的大小的任意数值---
            return 100;
        }
    }
```

(2) 按F11键在Android模拟器上执行应用程序。

(3) 单击Start Service按钮启动服务来下载文件。可以看到活动在Toast类显示Downloaded 100 bytes消息之前被“冻结”了几秒钟(如图10-3所示)。

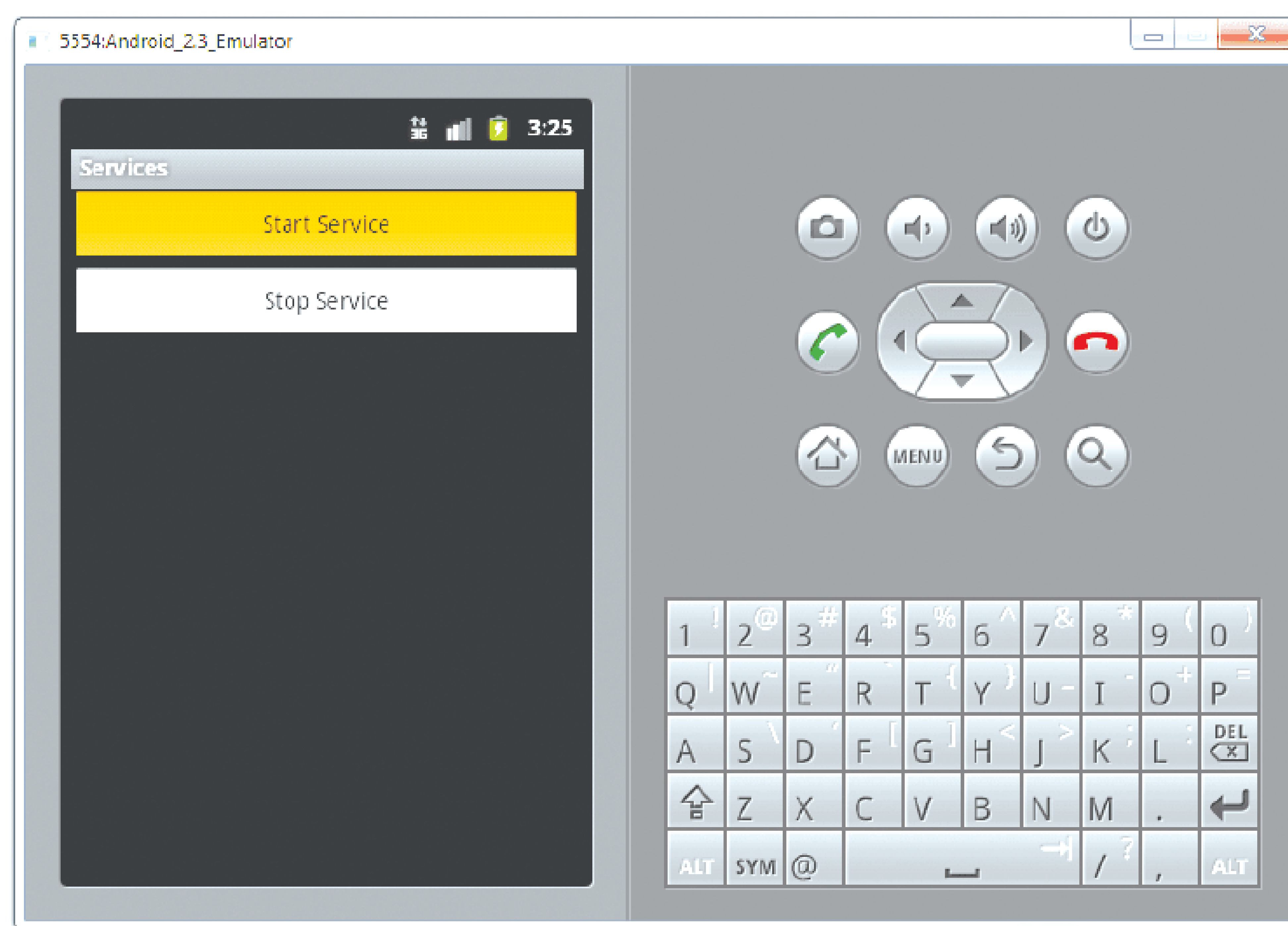


图 10-3

示例说明

在这个示例中，服务调用DownloadFile()方法来模拟从一个给定的URL下载文件。此方法返回下载的总字节数(已经硬编码为100)。为了模拟下载文件时服务所经历的延迟，使用Thread.Sleep()方法使服务暂停5秒(5000毫秒)。

当启动服务时，可以注意到活动被暂停了大约5秒钟，这正是从Internet上下载文件的时间。在这段时间内，整个活动没有响应，这证明了很重要的一点：服务和活动在相同的线程上运行。在这种情况下，因为服务暂停了5秒钟，所以活动也一样。

因此，对于一个长时间运行的服务，将所有长时间运行的代码放入一个单独的线程中是很重要的，这样服务就不会阻塞调用它的应用程序了。下面的“试一试”将告诉您如何做到这一点。

试一试　在服务中异步执行任务

(1) 使用在前一节中创建的同一个项目，在MyService.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Services;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;
import android.widget.Toast;

import java.net.MalformedURLException;
import java.net.URL;

import android.os.AsyncTask;

public class MyService extends Service {
    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // 我们希望这个服务在被显式停止前一直运行，所以返回“粘性的”状态。
        Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
        try {
            new DoBackgroundTask().execute(
                    new URL("http://www.amazon.com/somefiles.pdf"),
                    new URL("http://www.wrox.com/somefiles.pdf"),
                    new URL("http://www.google.com/somefiles.pdf"),
                    new URL("http://www.learn2develop.net/somefiles.pdf"));
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
        return START_STICKY;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Toast.makeText(this, "Service Destroyed", Toast.LENGTH_LONG).show();
    }

    private int DownloadFile(URL url) {
        try {
            //---模拟花一些时间来下载文件---
```

```
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            //---返回一个表示下载文件的大小的任意数值---
            return 100;
        }

        private class DoBackgroundTask extends AsyncTask<URL, Integer, Long> {
            protected Long doInBackground(URL... urls) {
                int count = urls.length;
                long totalBytesDownloaded = 0;
                for (int i = 0; i < count; i++) {
                    totalBytesDownloaded += DownloadFile(urls[i]);
                    //---计算下载的百分比并报告进度---
                    publishProgress((int) (((i+1) / (float) count) * 100));
                }
                return totalBytesDownloaded;
            }

            protected void onProgressUpdate(Integer... progress) {
                Log.d("Downloading files",
                        String.valueOf(progress[0]) + "% downloaded");
                Toast.makeText(getBaseContext(),
                    String.valueOf(progress[0]) + "% downloaded",
                    Toast.LENGTH_LONG).show();
            }

            protected void onPostExecute(Long result) {
                Toast.makeText(getBaseContext(),
                        "Downloaded " + result + " bytes",
                        Toast.LENGTH_LONG).show();
                stopSelf();
            }
        }
    }
```

(2) 按F11键在Android模拟器上调试应用程序。

(3) 单击Start Service按钮。Toast类将显示一个消息表明下载完成的百分比(如图10-4所示)。可以看到4个值：25%、50%、75%和100%。

(4) 还可以看到在LogCat窗口中的如下输出内容：

```
01-16 02:56:29.051: DEBUG/Downloading files(8844): 25% downloaded
01-16 02:56:34.071: DEBUG/Downloading files(8844): 50% downloaded
01-16 02:56:39.106: DEBUG/Downloading files(8844): 75% downloaded
01-16 02:56:44.173: DEBUG/Downloading files(8844): 100% downloaded
```

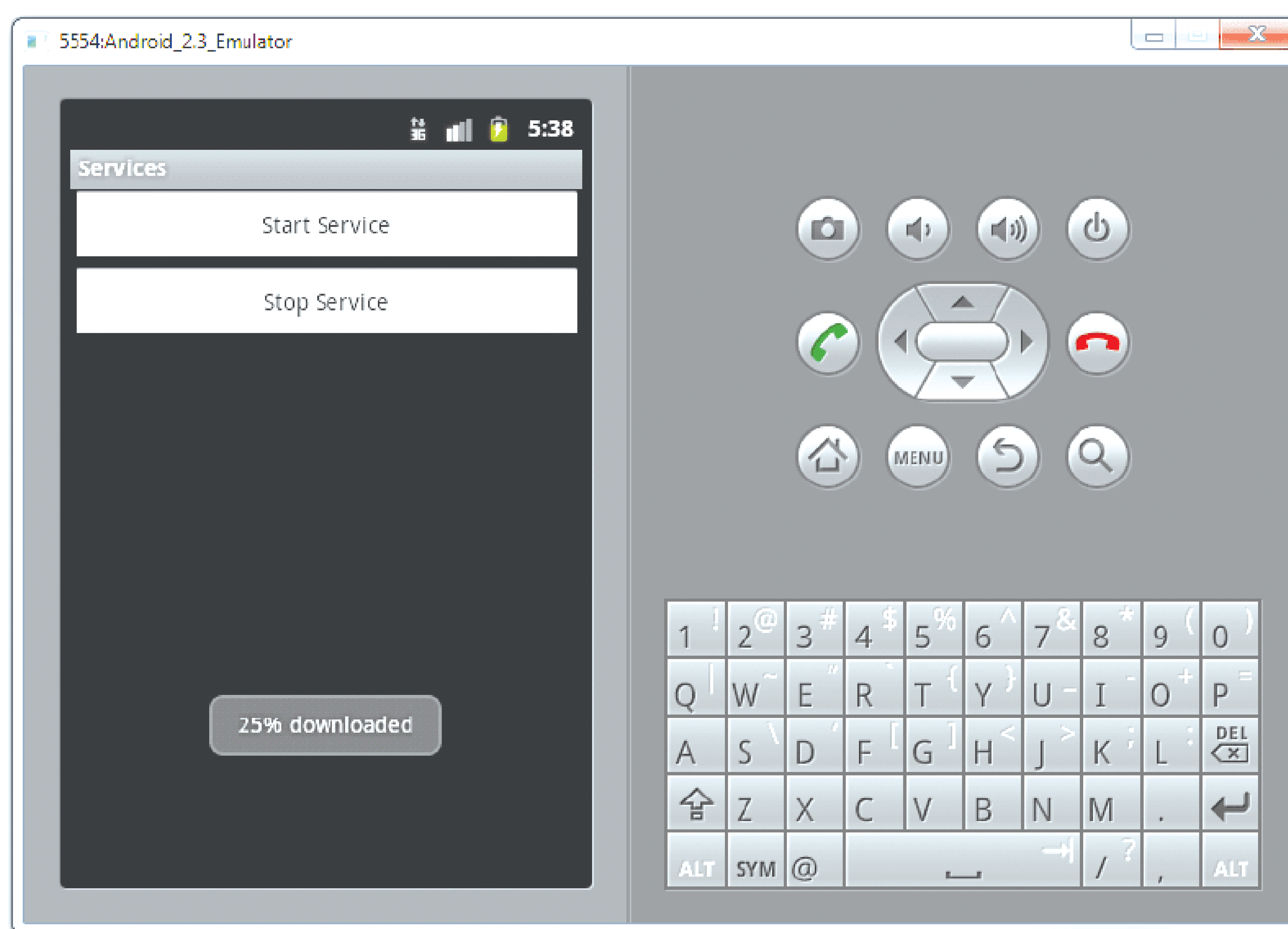


图 10-4

示例说明

这个示例说明了可以在您的服务中异步执行任务的一种方法。通过创建一个扩展AsyncTask类的内部类来做到这一点。AsyncTask类使您能够在后台执行，而无须手动操纵线程和处理程序。

DoBackgroundTask类通过指定3个泛型类型来扩展AsyncTask类：

```
private class DoBackgroundTask extends AsyncTask<URL, Integer, Long>{
```

这里指定的3种类型为URL、Integer和Long。这3种类型指定了以下3种方法所用到的数据类型，这些方法将在一个AsyncTask类中来实现：

- doInBackground()——这个方法接受一个先前指定的第一个泛型类型的数组。在这里，类型是URL。这个方法用于放置需要长时间运行的代码并在后台线程中执行。为了报告任务的进度，调用publishProgress()方法，它将调用第二个方法onProgressUpdate()，这个方法在一个AsyncTask类中实现。此方法的返回类型使用先前指定的第三个泛型类型，本例中是Long。
- onProgressUpdate()——这一方法在UI线程中启动并在调用publishProgress()方法时调用。它接受一个先前指定的第二个泛型类型的数组。在这里，类型是Integer。使用这一方法向用户报告后台任务的进度。
- onPostExecute()——这一方法在UI线程中启动并在doInBackground()方法执行完毕后调用。这一方法接受一个先前指定的第三个泛型类型的参数，本例中是Long。

要在后台下载多个文件，则创建DoBackgroundTask类的一个实例，然后通过传入一个URL数组来调用其execute()方法：

```
        try {
            new DoBackgroundTask().execute(
                    new URL("http://www.amazon.com/somefiles.pdf"),
                    new URL("http://www.wrox.com/somefiles.pdf"),
                    new URL("http://www.google.com/somefiles.pdf"),
```

```
                new URL("http://www.learn2develop.net/somefiles.pdf"));
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

前述的代码使服务在后台下载文件，并用文件下载的百分比形式报告进度。更重要的是，当在后台通过一个单独的线程下载文件时，活动仍然保持响应。

注意，当后台线程执行完毕时，需要手动调用stopSelf()方法来停止服务：

```
    protected void onPostExecute(Long result) {
        Toast.makeText(getBaseContext(),
                "Downloaded " + result + " bytes",
                Toast.LENGTH_LONG).show();
        stopSelf();
    }
```

stopSelf()方法相当于调用stopService()方法来停止服务。

10.1.2 在服务中执行重复的任务

除了在服务中执行长时间运行的任务，一些重复的任务也会在服务中执行。例如，您可能编写了一个一直在后台进行的闹钟服务。在这种情况下，服务可能需要定期执行一些代码来检查是否到了一个预设的时间以便发出提醒。要运行一个按固定的时间间隔执行的代码块，可以在服务中使用Timer类。下面的“试一试”将告诉您怎么做。

试一试 使用Timer类来运行重复的任务

(1) 使用在前一节中创建的同一个项目，在MyService.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Services;

import android.app.Service;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.IBinder;
import android.util.Log;
import android.widget.Toast;
import java.net.URL;

import java.util.Timer;
import java.util.TimerTask;

public class MyService extends Service {
    int counter = 0;
    static final int UPDATE_INTERVAL = 1000;
    private Timer timer = new Timer();
```

```
    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // 我们希望这个服务在被显式停止前一直运行，所以返回“粘性的”状态。
        Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
        doSomethingRepeatedly();
        return START_STICKY;
    }

    private void doSomethingRepeatedly() {
        timer.scheduleAtFixedRate( new TimerTask() {
            public void run() {
                Log.d("MyService", String.valueOf(++counter));
            }
        }, 0, UPDATE_INTERVAL);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        if (timer != null){
            timer.cancel();
        }
        Toast.makeText(this, "Service Destroyed", Toast.LENGTH_LONG).show();
    }
}
```

(2) 按F11键在Android模拟器上调试应用程序。

(3) 单击Start Service按钮。

(4) 观察在LogCat窗口中显示的输出内容：

```
01-16 15:12:04.364: DEBUG/MyService(495): 1
01-16 15:12:05.384: DEBUG/MyService(495): 2
01-16 15:12:06.386: DEBUG/MyService(495): 3
01-16 15:12:07.389: DEBUG/MyService(495): 4
01-16 15:12:08.364: DEBUG/MyService(495): 5
01-16 15:12:09.427: DEBUG/MyService(495): 6
01-16 15:12:10.374: DEBUG/MyService(495): 7
```

示例说明

在这一示例中，创建了一个Timer对象并在您已经定义的doSomethingRepeatedly()方法中调用这一对象的scheduleAtFixedRate()方法：

```
    private void doSomethingRepeatedly() {
```

```
        timer.scheduleAtFixedRate(new TimerTask() {
            public void run() {
                Log.d("MyService", String.valueOf(++counter));
            }
        }, 0, UPDATE_INTERVAL);
    }
```

将一个TimerTask类的实例传递给scheduleAtFixedRate()方法，以便可以重复执行位于run()方法中的代码块。scheduleAtFixedRate()方法的第二个参数指定了在第一次执行之前的时间间隔，以毫秒为单位。第三个参数以毫秒为单位，指定了后继执行之间的时间间隔。

在前面的示例中，基本上每1秒钟(1000毫秒)打印一次计数器的值。这一服务重复打印counter的值，直到服务终止。

```
    @Override
    public void onDestroy() {
        super.onDestroy();
        if (timer != null){
            timer.cancel();
        }
        Toast.makeText(this, "Service Destroyed", Toast.LENGTH_LONG).show();
    }
```

对于scheduleAtFixedRate()方法，不管每个任务需要多长时间，代码都是按照固定的时间间隔执行。例如，如果run()方法内的代码需要两秒钟来完成，那么第二个任务将在第一个任务结束后立即开始。同样，如果延迟时间设置为3秒，而完成任务只需要2秒，那么第二个任务在开始前将等待1秒钟。

10.1.3 使用IntentService在单独的线程上执行异步任务

本章的前面部分讲述了如何使用startService()方法启动服务以及使用stopService()方法停止服务。我们还学习了如何在一个单独的线程上执行长时间运行的任务——与主调活动不是同一个线程。重要的是需要注意，一旦服务执行完任务，应尽快停止，防止它继续占用宝贵的资源。这就是为什么当任务完成时需要使用stopSelf()方法来停止它的原因。遗憾的是，很多开发人员在任务已经完成后常常忘记将服务终止。为了能较容易地创建一个服务，使其异步运行一个任务并在结束时自行终止，可以使用IntentService类。

IntentService类是可以按需处理异步请求的Service的基类。它像一个普通服务那样启动，在一个工作者线程中执行其任务并在任务完成时自行终止。

下面的“试一试”演示了如何使用IntentService类。

试一试　使用IntentService类自动停止一个服务

(1) 使用前一节创建的同一个项目，添加一个新的类文件MyIntentService.java。

(2) 在MyIntentService.java文件中输入以下内容：

```
package net.learn2develop.Services;

import java.net.MalformedURLException;
import java.net.URL;

import android.app.IntentService;
import android.content.Intent;
import android.util.Log;

public class MyIntentService extends IntentService {

    public MyIntentService() {
        super("MyIntentServiceName");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        try {
            int result =
                DownloadFile(new URL("http://www.amazon.com/somefile.pdf"));
            Log.d("IntentService", "Downloaded " + result + " bytes");
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
    }

    private int DownloadFile(URL url) {
        try {
            //---模拟花一些时间来下载文件---
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return 100;
    }
}
```

(3) 在AndroidManifest.xml文件中添加下列粗体显示的语句：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Services"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
```

```
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".MyService" />
        <service android:name=".MyIntentService" />
    </application>
    <uses-sdk android:minSdkVersion="9" />
    <uses-permission android:name="android.permission.INTERNET"> </uses-permission>
</manifest>
```

(4) 在MainActivity.java文件中添加下列粗体显示的语句：

```
public class MainActivity extends Activity {
    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button btnStart = (Button) findViewById(R.id.btnStartService);
        btnStart.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //startService(new Intent(getBaseContext(), MyService.class));
                startService(new Intent(getBaseContext(), MyIntentService.class));
            }
        });

        Button btnStop = (Button) findViewById(R.id.btnStopService);
        btnStop.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                stopService(new Intent(getBaseContext(), MyService.class));
            }
        });
    }
}
```

(5) 按F11键在Android模拟器上调试应用程序。

(6) 单击Start Service按钮。大约5秒钟之后，应该可以在LogCat窗口中观察到以下语句：

```
01-17 03:05:21.244: DEBUG/IntentService(692): Downloaded 100 bytes
```

示例说明

首先，定义了MyIntentService类，它扩展IntentService类而非Service类：

```
public class MyIntentService extends IntentService {
    ...
}
```

需要实现这个类的构造函数并利用意图服务的名称(设置为一个字符串)调用其父类：

```
public MyIntentService() {
    super("MyIntentServiceName");
}
```

然后，实现onHandleIntent()方法，它在一个工作者线程上执行：

```
@Override
protected void onHandleIntent(Intent intent) {
    try {
        int result =
            DownloadFile(new URL("http://www.amazon.com/somefile.pdf"));
        Log.d("IntentService", "Downloaded " + result + " bytes");
    } catch (MalformedURLException e) {
        e.printStackTrace();
    }
}
```

需要在一个单独的线程中执行的代码可以放在onHandleIntent()方法中，如从服务器下载一个文件。当代码执行完毕，线程被终止，并且服务自动停止。

10.2　在服务和活动之间通信

服务通常只是在自己的线程中执行，独立于调用它的活动。如果您只是想让服务定期执行某些任务并且不需要将服务的状态通知给活动的话，这并不造成任何问题。例如，您可能有一个将设备的地理位置定期记录到数据库中的服务。在这种情况下，您的服务没有必要与任何活动进行交互，因为其主要目的是将坐标保存到数据库中。然而，假设您想监控一个特定的位置。当服务记录一个靠近您正在监控的位置的地址时，它可能需要将这一信息传递给活动。在这种情况下，就需要为服务和活动的交互设计一种方法。

下面的“试一试”展示了服务是如何使用一个BroadcastReceiver来与活动通信的。

试一试　从一个服务来启动一个活动

(1) 使用前一节创建的同一个项目，在MyIntentService.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Services;

import java.net.MalformedURLException;
import java.net.URL;
import android.app.IntentService;
import android.content.Intent;
import android.util.Log;
```

```
public class MyIntentService extends IntentService {
    public MyIntentService() {
        super("MyIntentServiceName");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        try {
            int result =
                DownloadFile(new URL("http://www.amazon.com/somefile.pdf"));
            Log.d("IntentService", "Downloaded " + result + " bytes");

            //---发送一个广播来通知活动文件已经被下载---
            Intent broadcastIntent = new Intent();
            broadcastIntent.setAction("FILE_DOWNLOADED_ACTION");
            getBaseContext().sendBroadcast(broadcastIntent);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
    }

    private int DownloadFile(URL url) {
        try {
            //---模拟花一些时间来下载文件---
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return 100;
    }
}
```

(2) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Services;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import android.content.IntentFilter;

public class MainActivity extends Activity {
    IntentFilter intentFilter;

    /** 当活动第一次被创建时调用。 */
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //---用来筛选文件已下载的意图---
        intentFilter = new IntentFilter();
        intentFilter.addAction("FILE_DOWNLOADED_ACTION");

        //---注册接收者---
        registerReceiver(intentReceiver, intentFilter);

        Button btnStart = (Button) findViewById(R.id.btnStartService);
        btnStart.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //startService(new Intent(getBaseContext(), MyService.class));
                startService(new Intent(getBaseContext(), MyIntentService.class));
            }
        });

        Button btnStop = (Button) findViewById(R.id.btnStopService);
        btnStop.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                stopService(new Intent(getBaseContext(), MyService.class));
            }
        });
    }

    private BroadcastReceiver intentReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            Toast.makeText(getBaseContext(), "File downloaded!",
                    Toast.LENGTH_LONG).show();
        }
    };
}
```

(3) 按F11键在Android模拟器上调试应用程序。

(4) 单击Start Service按钮。大约5秒钟后，Toast类将显示一个消息表明文件已经被下载(如图10-5所示)。

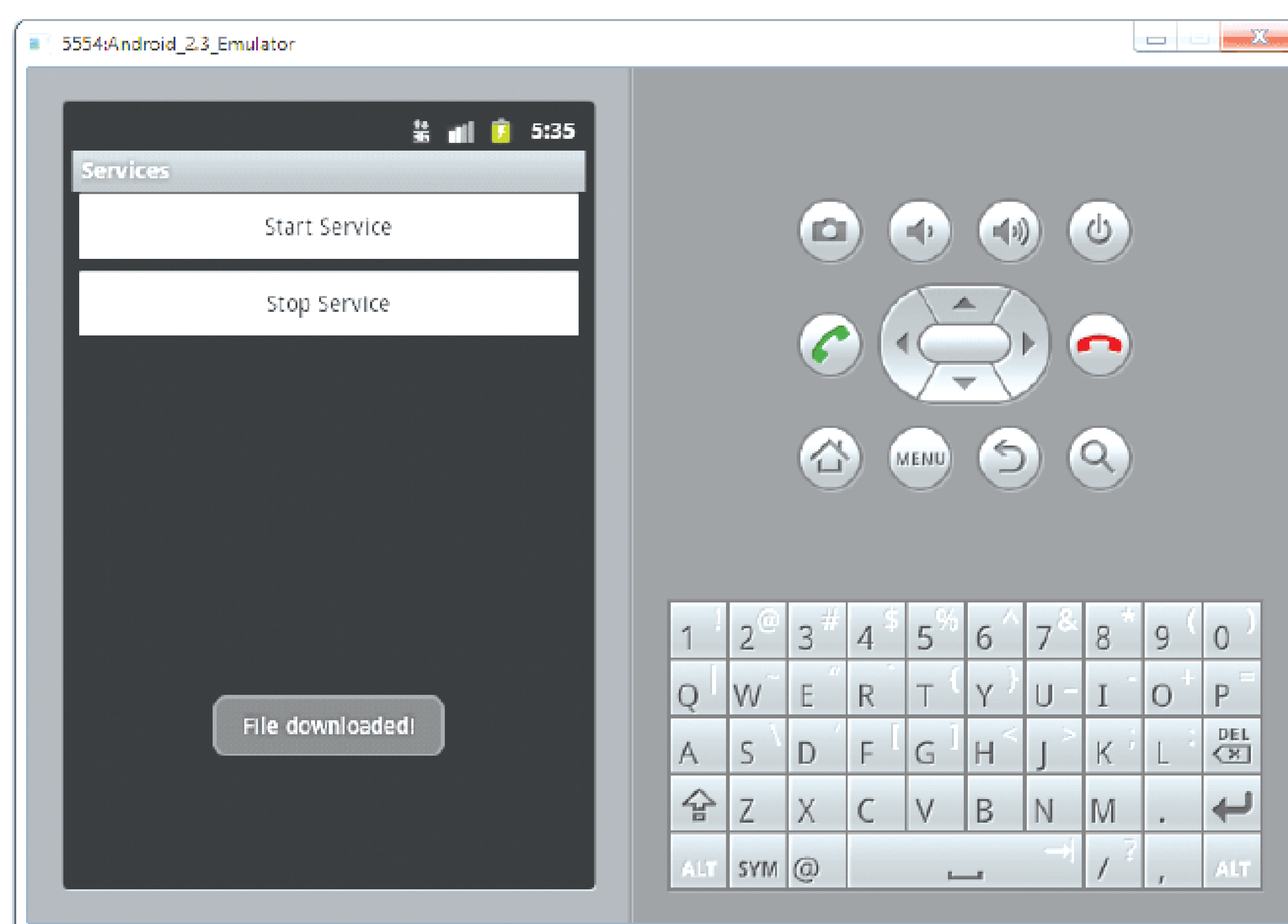


图 10-5

示例说明

为了在一个服务执行结束后可以通知一个活动，需要使用sendBroadcast()方法广播一个意图：

```
@Override
protected void onHandleIntent(Intent intent) {
    try {
        int result =
            DownloadFile(new URL("http://www.amazon.com/somefile.pdf"));
        Log.d("IntentService", "Downloaded " + result + " bytes");

        //---发送一个广播来通知活动文件已经被下载---
        Intent broadcastIntent = new Intent();
        broadcastIntent.setAction("FILE_DOWNLOADED_ACTION");
        getBaseContext().sendBroadcast(broadcastIntent);
    } catch (MalformedURLException e) {
        e.printStackTrace();
    }
}
```

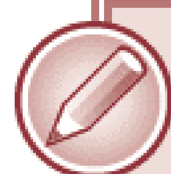
广播的这一意图的动作被设置为FILE_DOWNLOADED_ACTION，这意味着侦听这一意图的任何活动都将被调用。因此，在MainActivity.java文件中，使用IntentFilter类的registerReceiver()方法来侦听这个意图：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    //---用来筛选文件已下载的意图---
    intentFilter = new IntentFilter();
    intentFilter.addAction("FILE_DOWNLOADED_ACTION");
    //---注册接收者---
    registerReceiver(intentReceiver, intentFilter);
    ...
    ...
}
```

当收到这一意图后，它将启动一个已定义的BroadcastReceiver类的实例：

```
    private BroadcastReceiver intentReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            Toast.makeText(getBaseContext(), "File downloaded!",
                    Toast.LENGTH_LONG).show();
        }
    };
}
```



注意：第8章详细讨论了BroadcastReceiver类。

在本例中，显示了消息“File downloaded！”。当然，如果需要从服务传递一些数据给活动，可以使用Intent对象。10.3节将讨论这个问题。

10.3　将活动绑定到服务

到目前为止，您已经了解了如何创建服务，如何调用服务以及任务完成后如何终止服务。所有已经看到的服务都很简单——要么启动一个计数器并以固定的时间间隔递增，要么从Internet上下载一组固定的文件。然而，现实世界中的服务通常更为复杂，需要传递数据来使它们为您正确地工作。

使用先前展示的下载一组文件的服务，假设现在想让主调活动来决定下载什么样的文件，而不是在服务中靠硬编码实现，那么就需要按以下方式来做。

首先，在主调活动中，创建一个Intent对象，指定服务名称：

```
Button btnStart = (Button) findViewById(R.id.btnStartService);
btnStart.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(getBaseContext(), MyService.class);
    }
});
```

然后，创建一个URL对象的数组，并通过Intent对象的putExtra()方法将其赋给Intent对象。最后，使用Intent对象启动服务：

```
Button btnStart = (Button) findViewById(R.id.btnStartService);
btnStart.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(getBaseContext(), MyService.class);
        try {
            URL[] urls = new URL[] {
                    new URL("http://www.amazon.com/somefiles.pdf"),
                    new URL("http://www.wrox.com/somefiles.pdf"),
                    new URL("http://www.google.com/somefiles.pdf"),
                    new URL("http://www.learn2develop.net/somefiles.pdf")};
            intent.putExtra("URLs", urls);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
        startService(intent);
    }
});
```

注意，URL数组作为一个Object数组被赋给了Intent对象。

在服务端，需要通过在onStartCommand()方法中的Intent对象提取传入的数据：

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    // 我们希望这个服务在被显式停止前一直运行，所以返回“粘性的”状态.
    Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();

    Object[] objUrls = (Object[]) intent.getExtras().get("URLs");
    URL[] urls = new URL[objUrls.length];
    for (int i=0; i<objUrls.length-1; i++) {
        urls[i] = (URL) objUrls[i];
    }
    new DoBackgroundTask().execute(urls);
    return START_STICKY;
}
```

上述代码首先使用getExtras()方法提取数据来返回一个Bundle对象。然后使用get()方法以Object数组形式提取出URL数组。由于在Java中，不能直接将数组从一个类型转换到另一个类型，因此必须创建一个循环，对数组中的每个成员单独进行转换。最后，通过将URL数组传递给execute()方法来执行后台任务。

这是活动可以将值传递给服务的一种方式。正如您看到的，如果要将比较复杂的数据传递给服务，就必须做一些额外的工作，以确保数据被正确传递。传递数据的一个更好的办法是直接将活动绑定到服务上，这样活动可以直接调用服务的任何公共成员和方法。下面的“试一试”展示了如何将活动绑定到服务上。

试一试　直接通过绑定访问属性成员

(1) 使用先前创建的同一个项目，在MyService.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Services;

import java.net.URL;
import java.util.Timer;
import java.util.TimerTask;

import android.app.Service;
import android.content.Intent;
import android.os.AsyncTask;
import android.util.Log;
import android.widget.Toast;
import android.os.IBinder;

import android.os.Binder;

public class MyService extends Service {
    int counter = 0;
    URL[] urls;
```

```
    static final int UPDATE_INTERVAL = 1000;
    private Timer timer = new Timer();

    private final IBinder binder = new MyBinder();

    public class MyBinder extends Binder {
        MyService getService() {
            return MyService.this;
        }
    }

    @Override
    public IBinder onBind(Intent arg0) {
        //return null;
        return binder;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // 我们希望这个服务在被显式停止前一直运行，所以返回“粘性的”状态。
        Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
        new DoBackgroundTask().execute(urls);
        return START_STICKY;
    }

    @Override
    public void onDestroy() {
        ...
    }

    private int DownloadFile(URL url) {
        ...
    }

    private class DoBackgroundTask extends AsyncTask<URL, Integer, Long> {
        ...
    }
}
```

(2) 在MainActivity.java文件中添加下列粗体显示的语句：

```
package net.learn2develop.Services;

import java.net.MalformedURLException;
import java.net.URL;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
```

```
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import android.os.IBinder;
import android.content.ServiceConnection;

public class MainActivity extends Activity {
    IntentFilter intentFilter;

    private MyService serviceBinder;
    Intent i;

    private ServiceConnection connection = new ServiceConnection() {
        public void onServiceConnected(ComponentName className, IBinder service) {
            //---当建立连接时调用---
            serviceBinder = ((MyService.MyBinder)service).getService();
            try {
                URL[] urls = new URL[] {
                    new URL("http://www.amazon.com/somefiles.pdf"),
                    new URL("http://www.wrox.com/somefiles.pdf"),
                    new URL("http://www.google.com/somefiles.pdf"),
                    new URL("http://www.learn2develop.net/somefiles.pdf")};
                    //---通过serviceBinder对象将URL赋给服务---
                    serviceBinder.urls = urls;
                } catch (MalformedURLException e) {
                    e.printStackTrace();
                }
                startService(i);
        }
        public void onServiceDisconnected(ComponentName className) {
            //---当服务断开时调用---
            serviceBinder = null;
        }
    };

    /** 当活动第一次被创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //---用来筛选文件已下载的意图---
        intentFilter = new IntentFilter();
        intentFilter.addAction("FILE_DOWNLOADED_ACTION");

        //---注册接收者---
```

```
        registerReceiver(intentReceiver, intentFilter);

        Button btnStart = (Button) findViewById(R.id.btnStartService);
        btnStart.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                i = new Intent(MainActivity.this, MyService.class);
                bindService(i, connection, Context.BIND_AUTO_CREATE);
            }
        });

        Button btnStop = (Button) findViewById(R.id.btnStopService);
        btnStop.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                stopService(new Intent(getBaseContext(), MyService.class));
            }
        });
    }

    private BroadcastReceiver intentReceiver = new BroadcastReceiver() {
        ...
    };
}
```

(3) 按F11键调试应用程序。单击Start Service按钮将正常启动服务。

示例说明

为了将活动绑定到一个服务，必须首先在服务中声明一个扩展Binder类的内部类：

```
    public class MyBinder extends Binder {
        MyService getService() {
            return MyService.this;
        }
    }
```

在这个类中实现getService()方法，这一方法返回服务的一个实例。然后创建一个MyBinder类的实例：

```
    private final IBinder binder = new MyBinder();
```

还要修改onBind()方法来返回MyBinder实例：

```
    @Override
    public IBinder onBind(Intent arg0) {
        //return null;
        return binder;
    }
```

在onStartCommand()方法中，使用先前在服务中作为公共成员声明的urls数组调用execute()方法：

```
public class MyService extends Service {
```

```
        int counter = 0;
        URL[] urls;
    ...
    ...
    ...
        @Override
        public int onStartCommand(Intent intent, int flags, int startId) {
            // 我们希望这个服务在被显式停止前一直运行，所以返回“粘性的”状态。
            Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
            new DoBackgroundTask().execute(urls);
            return START_STICKY;
        }
```

下一步要做的就是从活动中直接对这个URL数组进行设置。

在MainActivity.java文件中，首先声明服务的一个实例和一个Intent对象：

```
        private MyService serviceBinder;
        Intent i;
```

serviceBinder对象将用作指向服务的引用，可以直接访问它。

然后创建一个ServiceConnection类的实例以便监控服务的状态：

```
        private ServiceConnection connection = new ServiceConnection() {
            public void onServiceConnected(ComponentName className, IBinder service) {
                //---当建立连接时调用---
                serviceBinder = ((MyService.MyBinder)service).getService();
                try {
                    URL[] urls = new URL[] {
                        new URL("http://www.amazon.com/somefiles.pdf"),
                        new URL("http://www.wrox.com/somefiles.pdf"),
                        new URL("http://www.google.com/somefiles.pdf"),
                        new URL("http://www.learn2develop.net/somefiles.pdf")};
                        //---通过serviceBinder对象将URL赋给服务---
                        serviceBinder.urls = urls;
                    } catch (MalformedURLException e) {
                        e.printStackTrace();
                    }
                    startService(i);
            }
            public void onServiceDisconnected(ComponentName className) {
                //---当服务断开时调用---
                serviceBinder = null;
            }
        };
```

需要实现两个方法：onServiceConnected()和onServiceDisconnected()。onServiceConnected()方法是当活动连接到服务时调用的。当服务与活动断开时调用onServiceDisconnected()方法。

在onServiceConnected()方法中，当活动连接到服务时，通过使用service参数的getService()方法，然后将其赋给serviceBinder对象来获取服务的一个实例。serviceBinder对象是一个指向服务

的引用，可以通过这一对象访问服务的所有成员和方法。这里，创建了一个URL数组并将其直接赋给服务中的公共成员：

```
URL[] urls = new URL[] {
    new URL("http://www.amazon.com/somefiles.pdf"),
    new URL("http://www.wrox.com/somefiles.pdf"),
    new URL("http://www.google.com/somefiles.pdf"),
    new URL("http://www.learn2develop.net/somefiles.pdf")};

    //---通过serviceBinder对象将URL赋给服务---
    serviceBinder.urls = urls;
```

然后使用一个Intent对象启动服务：

```
startService(i);
```

在可以启动服务之前，需要将活动绑定到服务。这个在Start Service按钮的onClick()方法中完成：

```
Button btnStart = (Button) findViewById(R.id.btnStartService);
btnStart.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        i = new Intent(MainActivity.this, MyService.class);
        bindService(i, connection, Context.BIND_AUTO_CREATE);
    }
});
```

bindService()方法使活动与服务建立了连接。它接受3个参数：一个Inent对象、一个ServiceConnection对象以及一个用来指示服务绑定方式的标志。

10.4　本章小结

在本章中，我们学习了如何在Android应用程序中创建一个服务来执行长时间运行的任务。了解了可以用来确保后台任务以异步方式执行而不阻塞主调活动的许多方法。我们还学习了活动是如何给服务传递数据的，以及如何绑定到一个活动，使之可以更直接地访问服务。

练　习

1. 为什么说将一个服务中的长时间运行的代码放在一个单独的线程中是重要的?
2. IntentService类的作用是什么?
3. 说出需要在一个AsyncTask类中实现的3个方法。
4. 服务如何通知活动发生了一个事件?

练习答案参见附录C。

本章主要内容

主　题	关 键 概 念
创建一个服务	创建一个类并扩展Service类
在一个服务中实现方法	实现以下方法：onBind()、onStartCommand()和onDestroy()
启动一个服务	使用startService()方法
停止一个服务	使用stopService()方法
执行长时间运行的任务	使用AsyncTask类并实现3个方法：doInBackground()、onProgressUpdate()和onPostExecute()
执行重复的任务	使用Timer类并调用它的scheduleAtFixedRate()方法
在一个单独的线程中执行任务并自动停止一个服务	使用IntentService类
在活动和服务之间通信	使用Intent对象给服务传递数据。对于一个服务，广播一个Intent来通知一个活动
将活动绑定到服务	在服务中使用Binder类并在主调活动中实现ServiceConnection类

第11章

发布Android应用程序

本章将介绍以下内容

- 如何为部署应用程序做准备
- 如何将应用程序导出为一个APK文件并用新的证书对其签名
- 如何分发Android应用程序
- 如何在Android Market上发布应用程序

到目前为止，您已经了解到了使用Android可以做很多有趣的事情。然而，为了使您的应用程序可以在用户的设备上运行，需要一个方法来部署和分发它。在本章中，将学习如何为部署Android应用程序做准备并将它们转移到客户设备上。此外，还将学习如何将您的应用程序发布到Android Market上，在那里您可以通过出售应用程序来赚钱！

11.1　为发布做准备

Google已经使得Android应用程序的发布变得相当容易，因此可以很迅速地将其分发给终端用户。发布Android应用程序的步骤通常包含以下几步：

(1) 将应用程序导出为一个APK(Android Package)文件。

(2) 生成自己的自签名证书并用它对应用程序进行数字签名。

(3) 部署签名后的应用程序。

(4) 利用Android Market对应用程序进行托管和出售。

在接下来的章节中，将学习如何为签署应用程序做准备，以及学习部署应用程序的不同方法。

本章中将使用在第9章创建的LBS项目来说明如何部署Android应用程序。

11.1.1　版本化

从Android SDK 1.0版本开始，每一个Android应用程序的AndroidManifest.xml文件都包括了android:versionCode和android:versionName属性：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.LBS"
      android:versionCode="1"
```

```
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
    <uses-library android:name="com.google.android.maps" />
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="7" />
    <uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
</manifest>
```

android:versionCode属性表示了应用程序的版本号。对于对应用程序所做的每一次修改，都应该将这个值加1，以便可以以编程方式来区分先前版本和最新版本。Android系统永远不会使用这个值。但对于开发来说，这是一个获得应用程序版本号的很有用的方法。不过，Android Market使用android:versionCode属性来确定应用程序是否有新的版本可用。

可以使用PackageManager类的getPackageInfo()方法以编程方式检索android:versionCode属性的值，如下所示：

```
PackageManager pm = getPackageManager();
try {
    //---获取包的信息---
    PackageInfo pi =
        pm.getPackageInfo("net.learn2develop.LBS", 0);
    //---显示版本代码---
    Toast.makeText(getBaseContext(),
        "VersionCode: " +Integer.toString(pi.versionCode),
        Toast.LENGTH_SHORT).show();
} catch (NameNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

android:versionName属性包含了对用户可见的版本信息。它应该以<*major*>.<*minor*>.<*point*>格式包含值。如果应用程序进行了重要的升级，那么应该将<*major*>加1。对于微小的增量更新，可以将<*minor*>或<*point*>加1。例如，一个新的应用程序的版本名称是1.0.0。对于较小的增量更新，可以将版本名称修改为1.1.0或1.0.1。如果下一次有较大的更新，可以将其变为2.0.0。

如果计划在Android Market(www.android.com/market/)上发布应用程序，AndroidManifest.xml文件必须具有以下属性：

- android:versionCode(位于<manifest>元素中)
- android:versionName(位于<manifest>元素中)
- android:icon(位于<applicationt>元素中)

● android:label(位于<application>元素中)

android:label属性指定了应用程序的名称。这个名称将显示在Android设备的Settings | Applications | Manage Application部分中。对于LBS项目，我们给应用程序起名为Where Am I:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.LBS"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="Where Am I">
    <uses-library android:name="com.google.android.maps" />
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="7" />
    <uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
</manifest>
```

另外，如果应用程序需要一个最低版本的SDK，那么可以在AndroidManifest.xml文件中使用<uses-sdk>元素来指定：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.LBS"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="Where Am I">
    <uses-library android:name="com.google.android.maps" />

        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="7" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
</manifest>
```

在前面的示例中，应用程序需要的最低SDK版本为7，即Android 2.1。一般来说，将这个版本号设置为应用程序可支持的最低版本是明智的。这可以保证有更广泛的用户可以运行您的应用程序。尽管在写作本书时Android的最新版本是2.3，但是仍旧有大量设备在运行Android 2.1和2.2。

11.1.2 对Android应用程序进行数字签名

所有Android应用程序在被允许部署到设备(或模拟器)上之前必须经过数字签名。与一些手机平台不同，您不需要从认证机构(CA)购买数字证书来进行签名。相反，您可以生成自己的自签名证书并用它为应用程序签名。

如果使用Eclipse开发Android应用程序，然后按F11键将其部署到一个模拟器上，Eclipse将自动为您对其进行签名。可以在Eclipse中选择Window | Preferences，展开Android项，选择Build(如图11-1所示)来验证这一点。Eclipse使用一个默认的调试密钥库(被命名为debug.keystore)来对应用程序签名。密钥库常被称为数字证书。

如果您正在发布一个Android应用程序，就必须使用您自己的证书来对其进行签名。使用调试证书签名的应用程序是不能被发布的。虽然您可以使用Java SDK提供的keytool.exe实用程序来手动生成自己的证书，但Eclipse通过提供一个向导使这一过程变得更容易了，它可以引导您一步步地生成证书。它将使用生成的证书对应用程序进行签名(也可以使用Java SDK的jarsigner.exe工具进行手动签名)。

下面的“试一试”展示了如何使用Eclipse导出一个Android应用程序并使用新生成的证书进行签名。

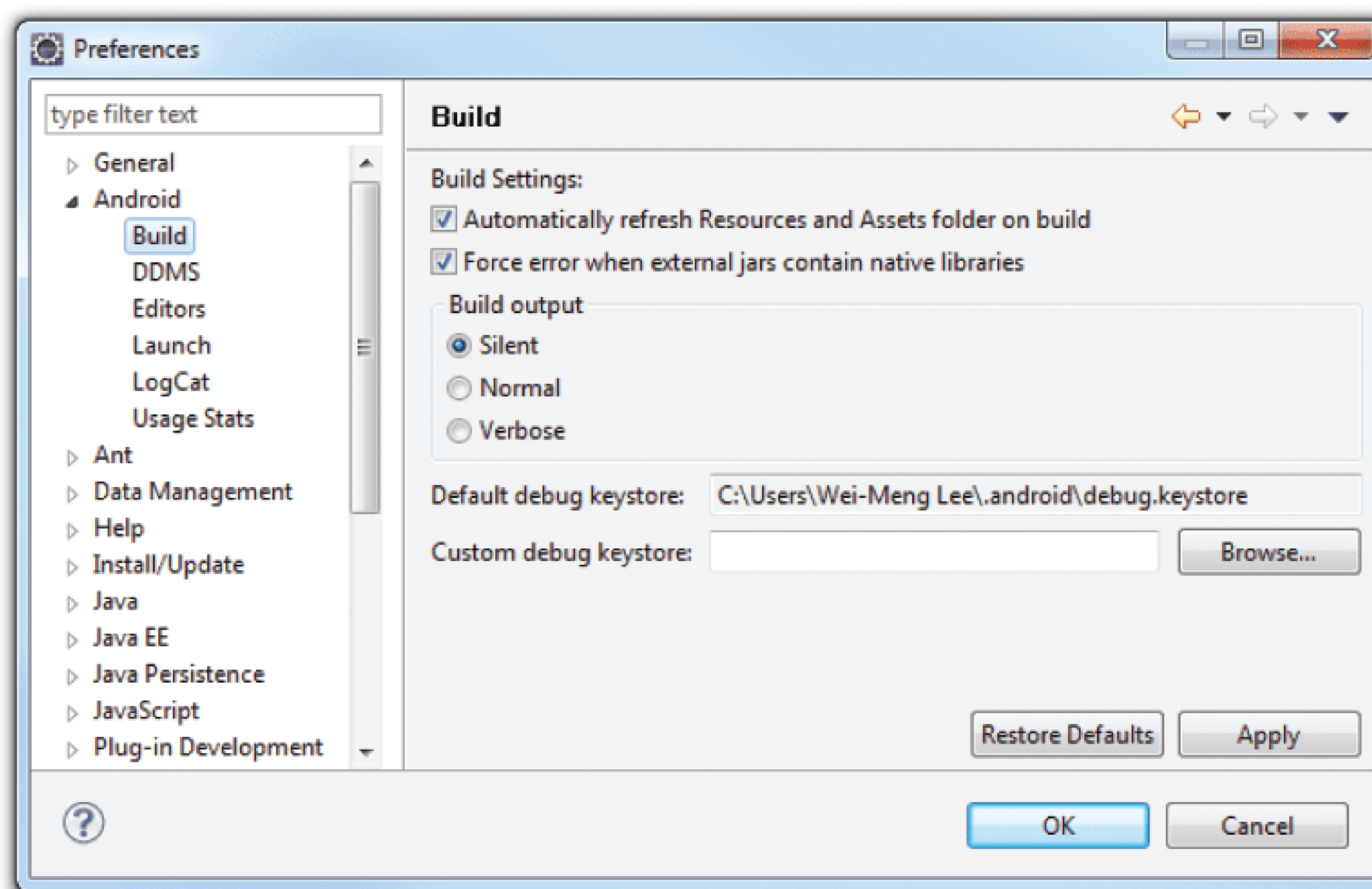


图 11-1

试一试　导出Android应用程序并对其签名

(1) 启动Eclipse，打开在第9章创建的LBS项目。

(2) 在Eclipse中选择LBS项目并选择File | Export...。

(3) 在Export对话框中，展开Android项，并选择Export Android Application(如图11-2所示)。

(4) 现在LBS项目应该显示出来了(如图11-3所示)，单击Next按钮。

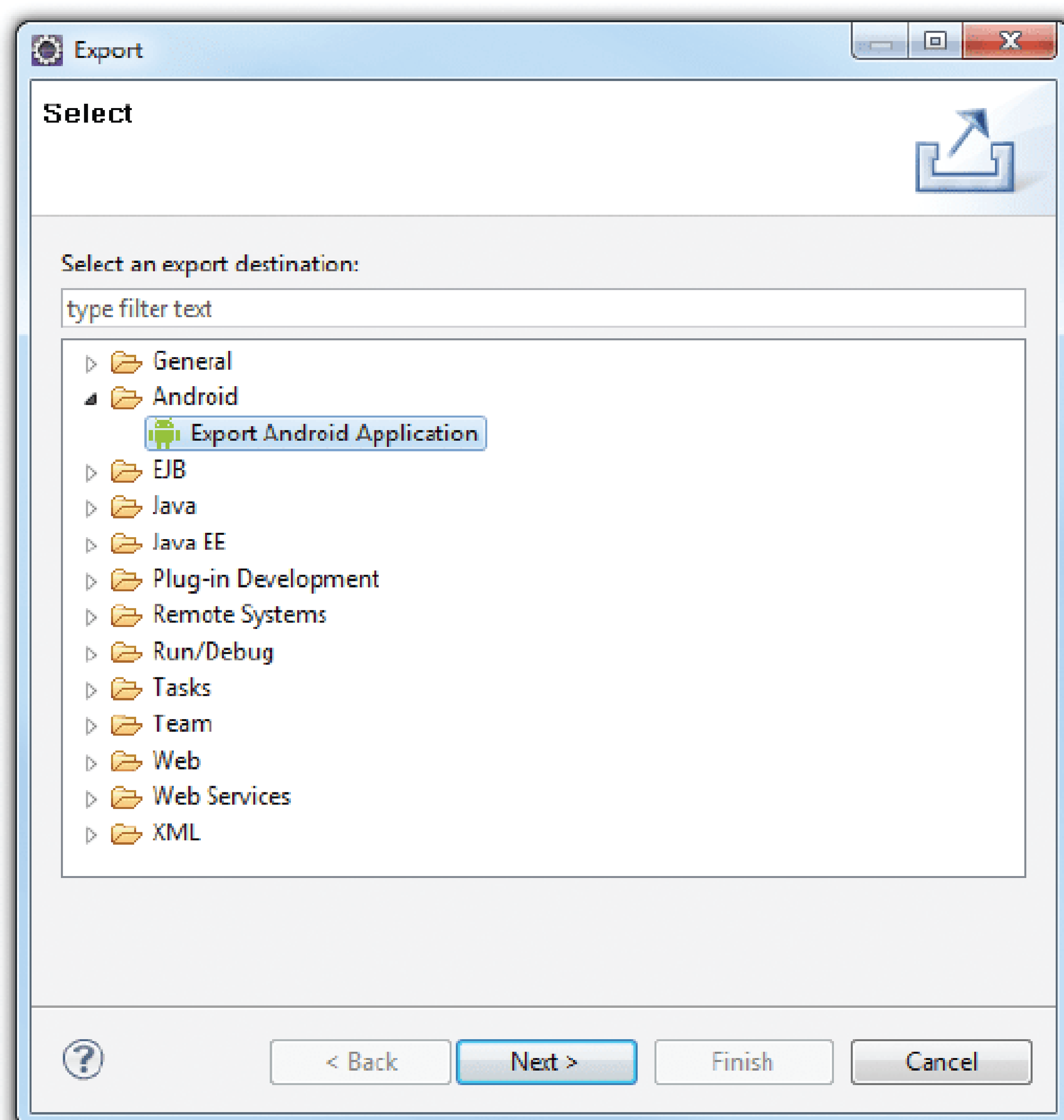


图 11-2

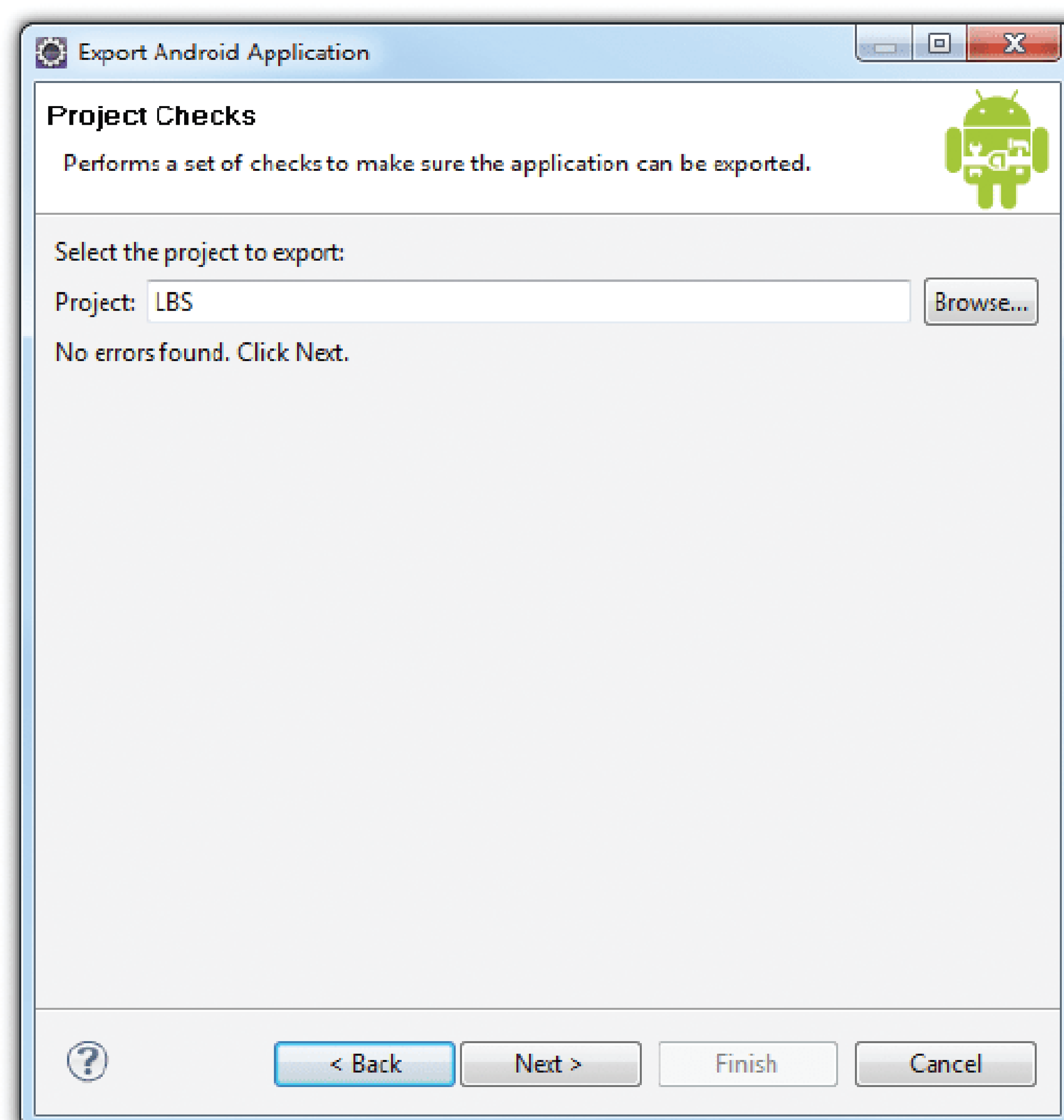


图 11-3

(5) 选择Create new keystore选项来创建一个新的证书(密钥库)用于应用程序签名(如图11-4所示)。输入保存新密钥库的路径并输入一个密码来保护此密钥库。在本例中，输入password作为密码，单击Next按钮。

(6) 为私钥提供一个别名(命名为DistributionKeyStoreAlias，如图11-5所示)并输入一个密码来保护私钥。在本例中，输入password作为密码。还需要输入密钥的有效期。根据Google的规定，您的应用程序必须用一个加密的私钥进行签名，其有效期要到2033年10月22日以后才能结束。因此，应该输入一个大于2033减去当前年份的数。单击Next按钮。

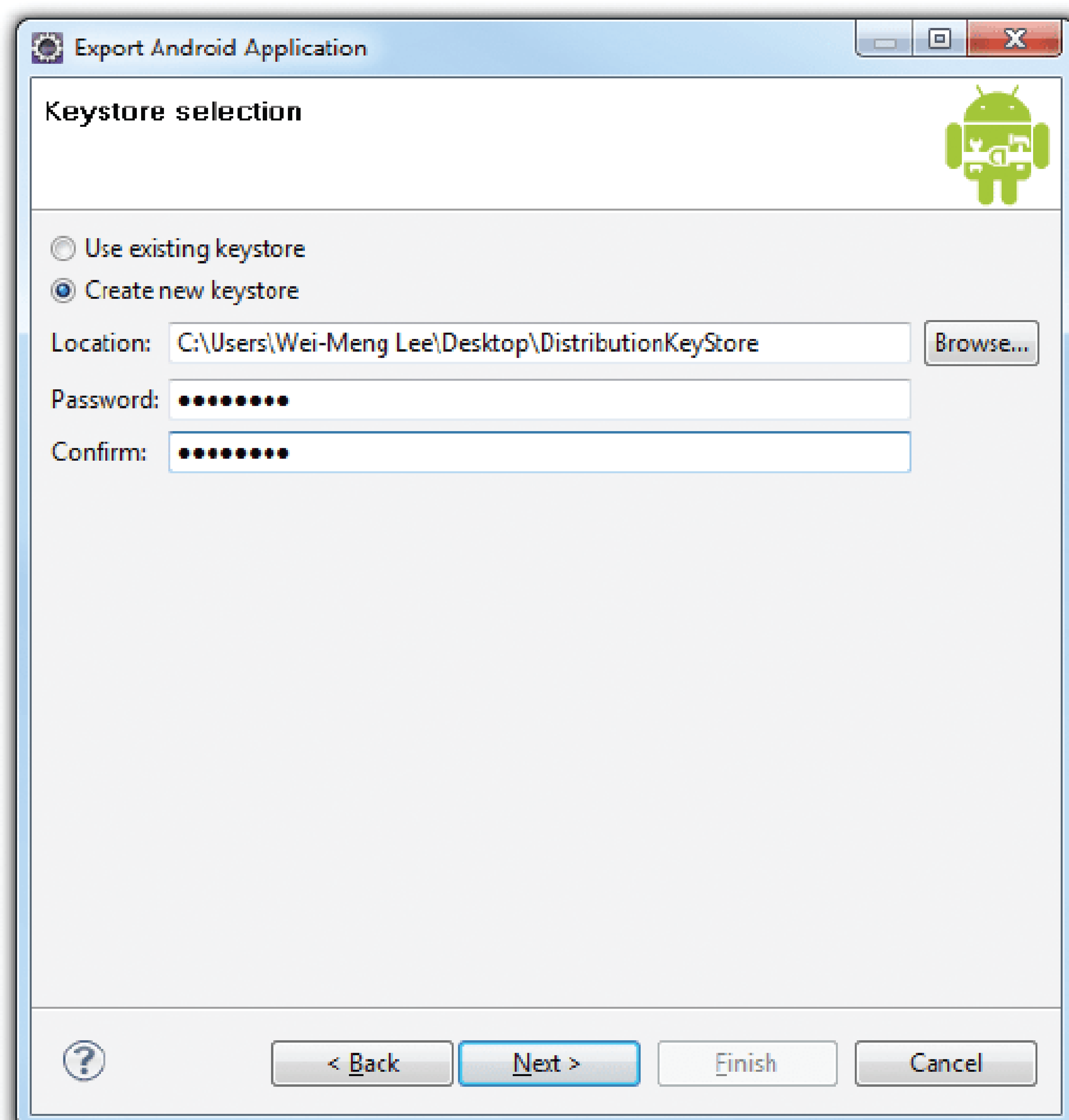


图 11-4

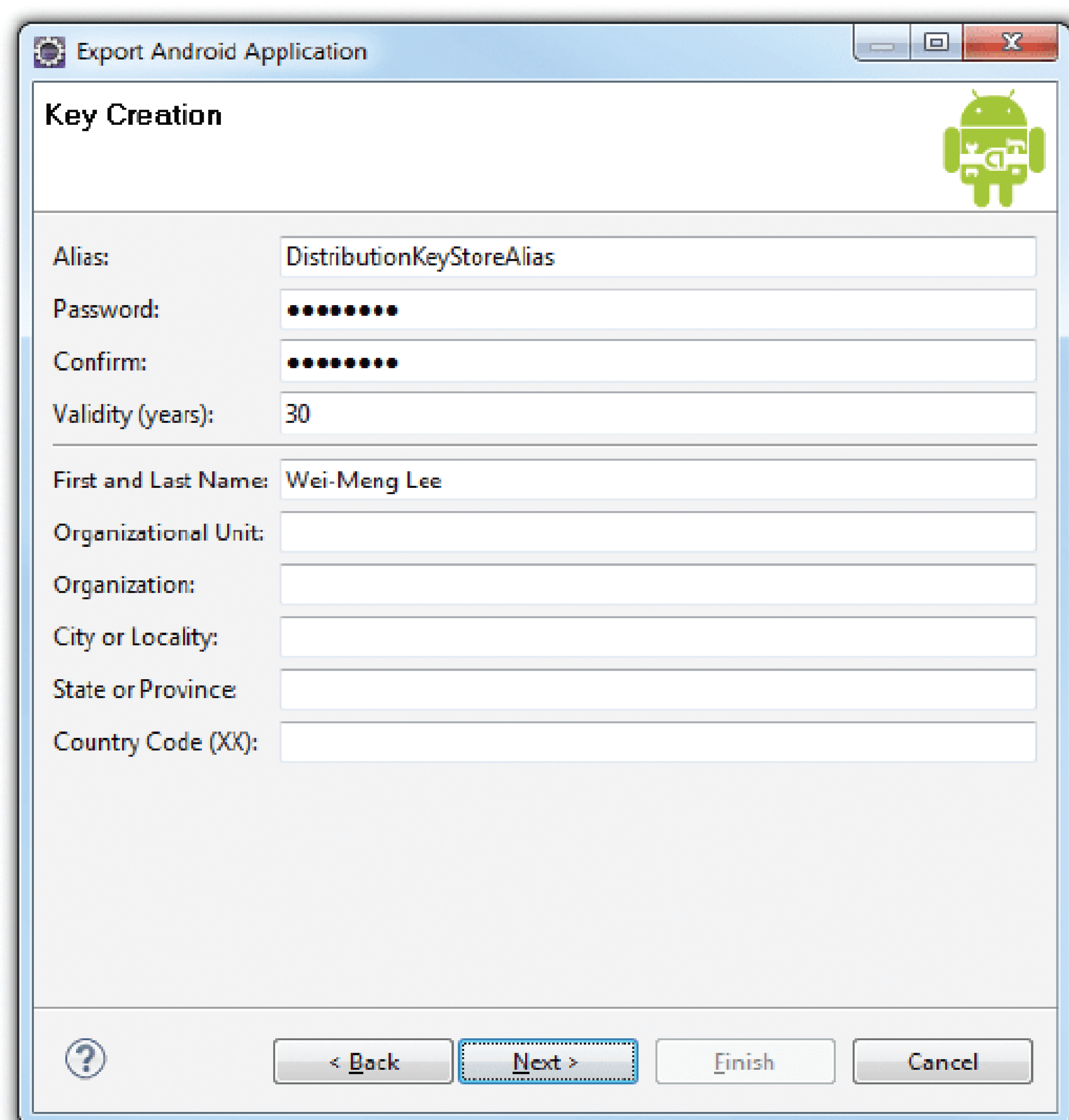


图 11-5

(7) 输入存储APK目标文件的路径(如图11-6所示)。单击Finish按钮，将生成APK文件。

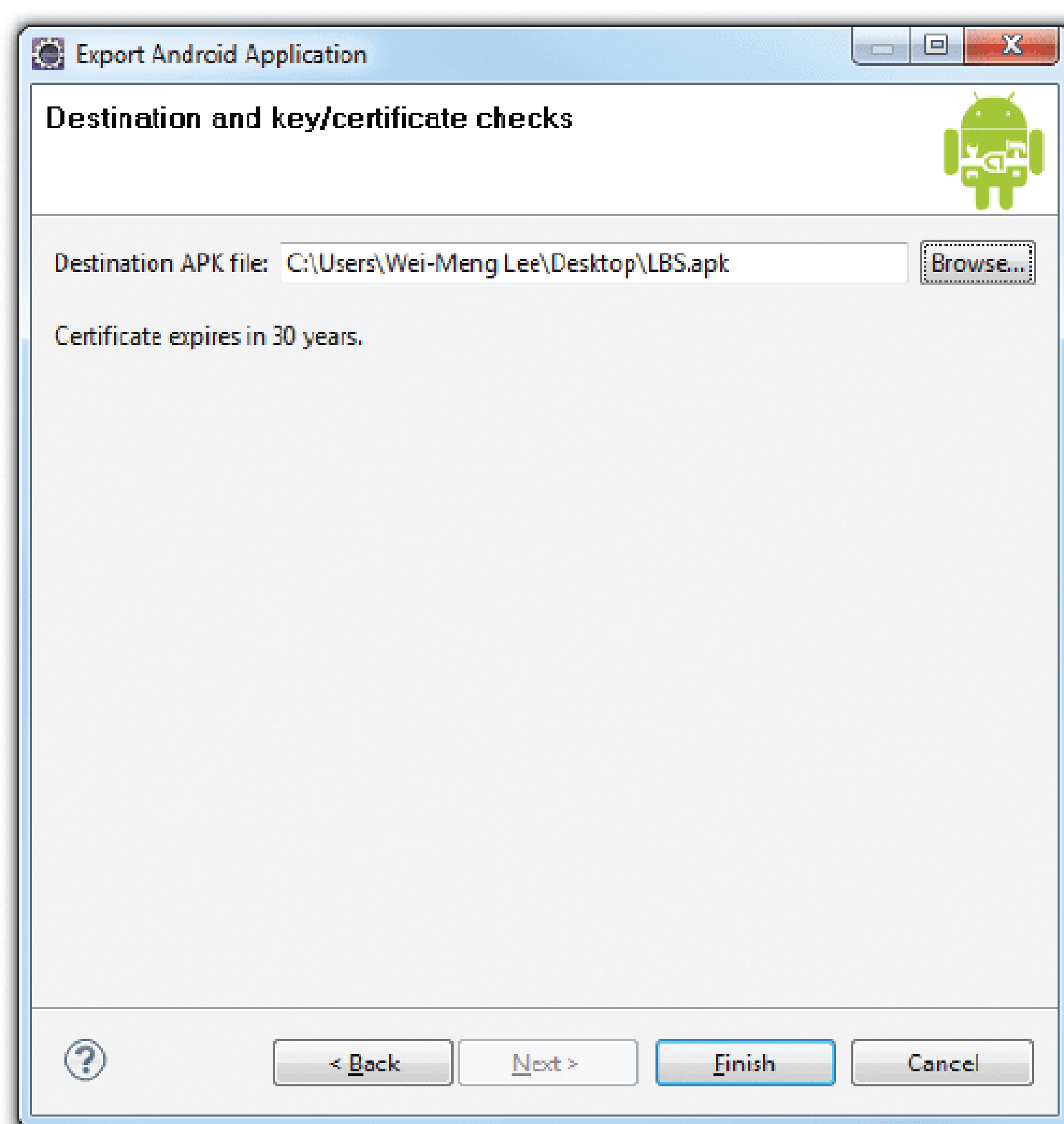


图 11-6

(8) 在第9章中，LBS应用程序需要使用Google Maps API密钥。这个密钥是通过使用您的debug.keystore的MD5指纹申请到的。这意味着Google Maps API密钥与用来对应用程序进行签名的debug.keystore是捆绑在一起的。因为您现在生成了新的密钥库来对应用程序进行签名和部署，所以需要使用新密钥库的MD5指纹再次申请Google Maps API密钥。要做到这一点，在命令提示符窗口中输入以下命令(keytool.exe实用程序的位置可能略微不同，需要使用在前面第(5)步所选择的路径来替换这个密钥库的路径，如图11-7所示)：

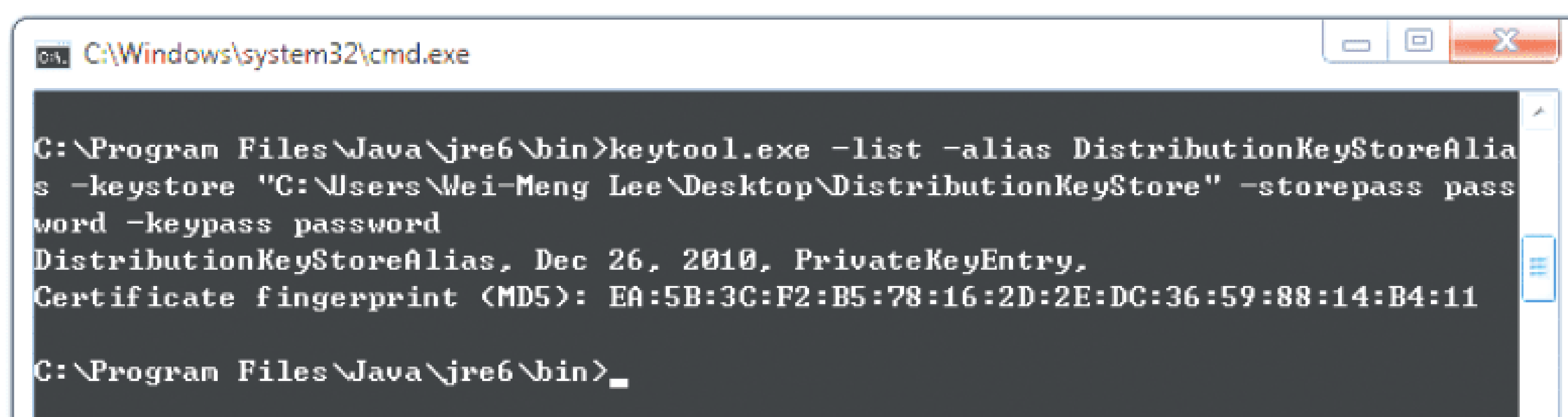


图 11-7

```
C:\Program Files\Java\jre6\bin>keytool.exe -list -alias
DistributionKeyStoreAlias -keystore "C:\Users\Wei-Meng Lee\Desktop\
DistributionKeyStore" -storepass password -keypas spassword
```

(9) 使用从前面的步骤获得的MD5指纹，转到http://code.google.com/android/add-ons/google-apis/maps-api-signup.html并签署一个新的Maps API密钥。

(10) 在main.xml文件中输入新的Maps API密钥：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<com.google.android.maps.MapView
```

```
        android:id="@+id/mapView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:enabled="true"
        android:clickable="true"
        android:apiKey="<Your Key Here>" />
</LinearLayout>
```

(11) 由于在main.xml文件中输入了新的Maps API密钥，现在需要您再一次导出应用程序并重新对其签名。重复第2～4步。当要求您选择一个密钥库时，选择Use existing keystore选项(如图11-8所示)并输入一个先前用于保护您的密钥库的密码(在这里，密码是password)。单击Next按钮。

(12) 选择Use existing key选项(如图11-9所示)并输入先前设置的用于保护私钥的密码(输入password)。单击Next按钮。

(13) 单击Finish按钮(如图11-10所示)再次生成APK文件。

到此为止，APK已经生成了，它包含了捆绑到新密钥库的新的Maps API密钥。

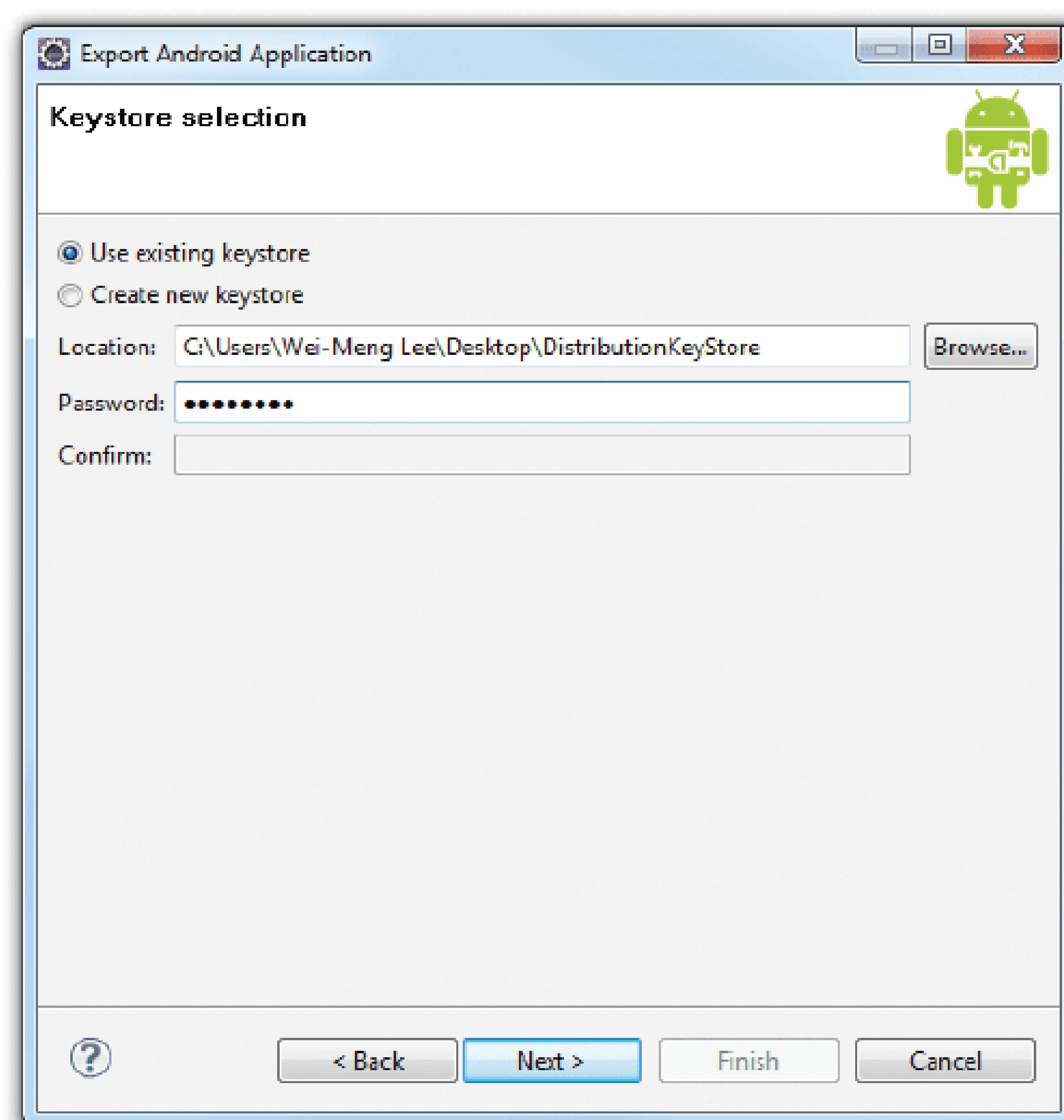


图 11-8

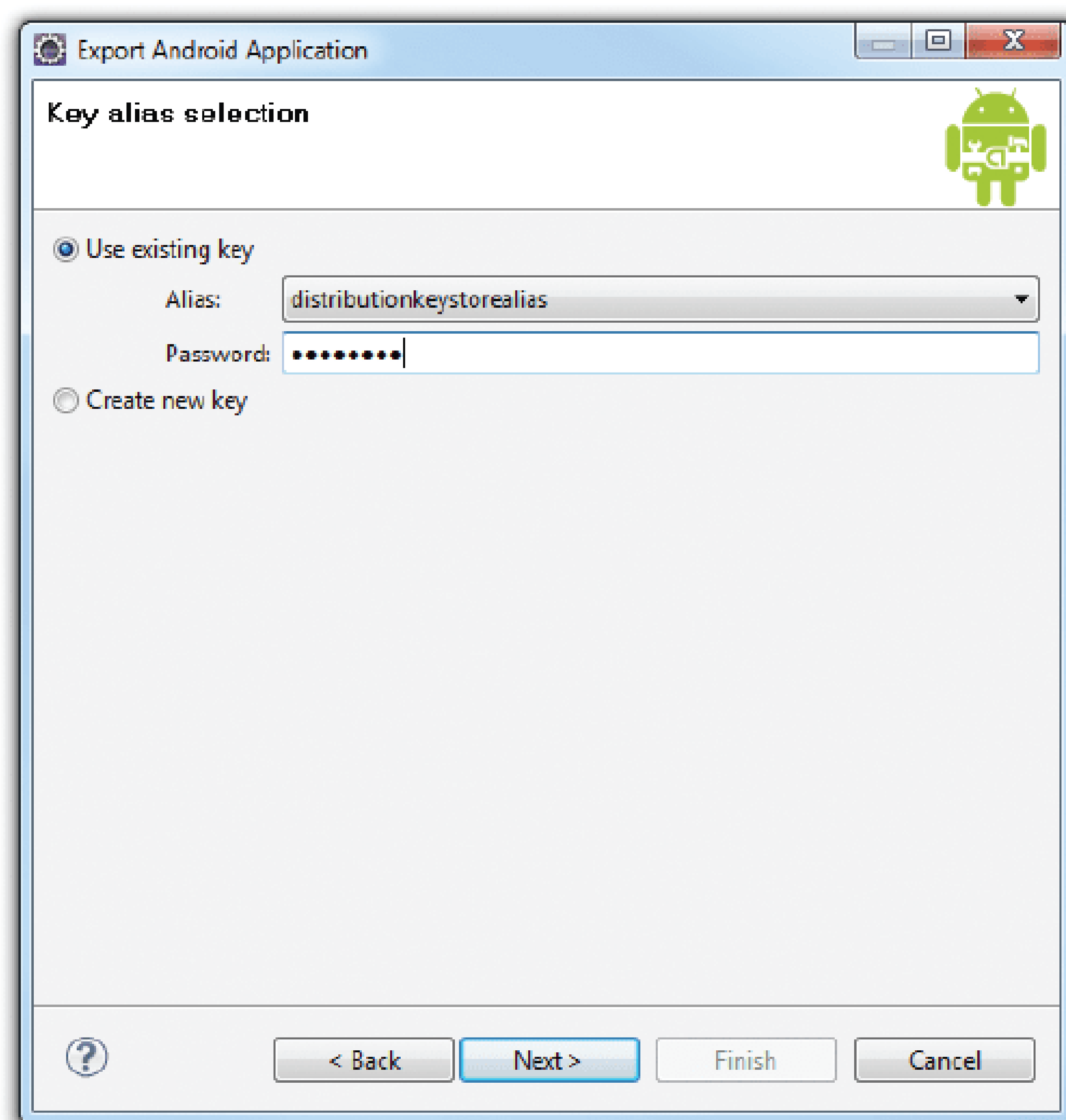


图 11-9

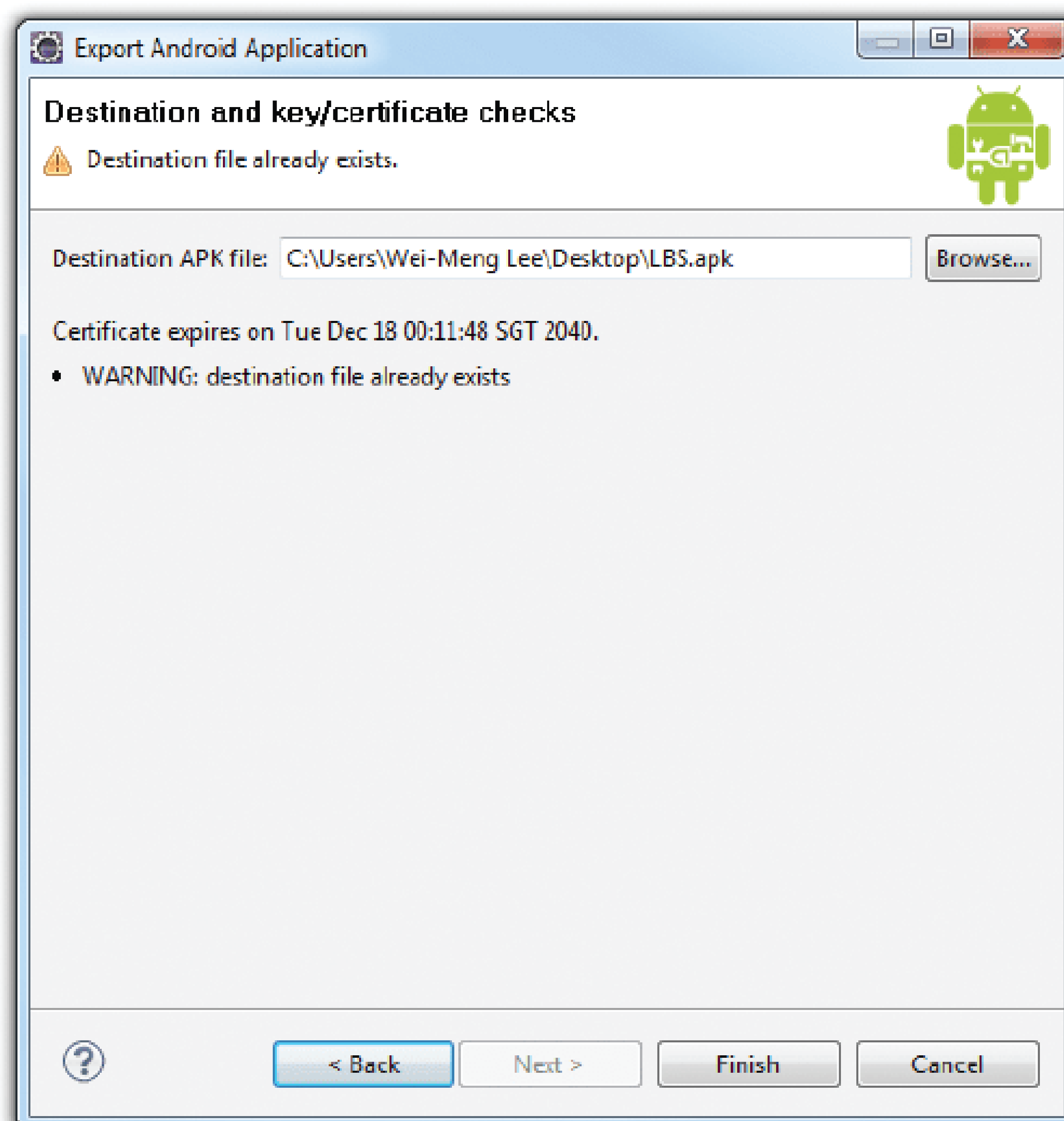


图 11-10

示例说明

Eclipse提供了Export Android Application选项，可以帮助您将Android应用程序导出为一个APK文件并生成一个用于对APK文件进行签名的新的密钥库。对于使用Maps API的应用程序，要注意Maps API必须和为您的APK文件签名的新密钥库相关联。

11.2 部署APK文件

一旦对APK文件进行了签名，就需要有个方法将它们转移到用户设备上。下面的小节将描述部署APK文件的不同方法，主要包括以下3个方法：

- 使用adb.exe工具手动部署
- 在Web服务器上托管应用程序
- 通过Android Market发布

除了以上方法，还可以通过电子邮件、SD卡等方式将您的应用程序安装到用户设备上。只要能将APK文件转移到用户设备上，您就能够安装这些应用程序。

11.2.1 使用adb.exe工具

一旦Android应用程序经过签名，就可以使用abd.exe(Android Debug Bridge)工具(位于Android SDK的platform-tools文件夹中)将其部署到模拟器和设备上。

使用Windows中的命令提示符，转到<Android_SDK>\platform-tools文件夹下。要在模拟器/设备上安装应用程序(假设模拟器当前已经启动并运行或设备已连接)，可输入以下命令：

```
adb install "C:\Users\Wei-Meng Lee\Desktop\LBS.apk"
```

adb.exe工具介绍

adb.exe工具是一个多功能的工具，可以用来控制与您的计算机相连的Android设备(和模拟器)。

默认情况下，当使用adb命令时，假定当前只连接了一台设备/模拟器。如果连接了多台设备，那么adb命令会返回一个错误消息：

```
error: more than one device and emulator
```

可以使用adb的device选项来查看当前连接到计算机上的设备，如下所示：

```
D:\Android 2.3\android-sdk-windows\platform-tools>adb devices
List of devices         attached
HT07YPY09335            device
emulator-5554           device
emulator-5556           device
```

正如上面的示例所显示的，这个命令返回一个当前已连接设备的列表。为了给一个特定的设备发出命令，需要使用-s选项来指明此设备，如下所示：

```
adb -s emulator-5556 install LBS.apk
```

如果试图在一台已有APK文件的设备上安装该文件，将显示以下错误消息：

```
Failure [INSTALL_FAILED_ALREADY_EXISTS]
```

图11-11展示了在一台真实设备上成功安装了一个APK文件。

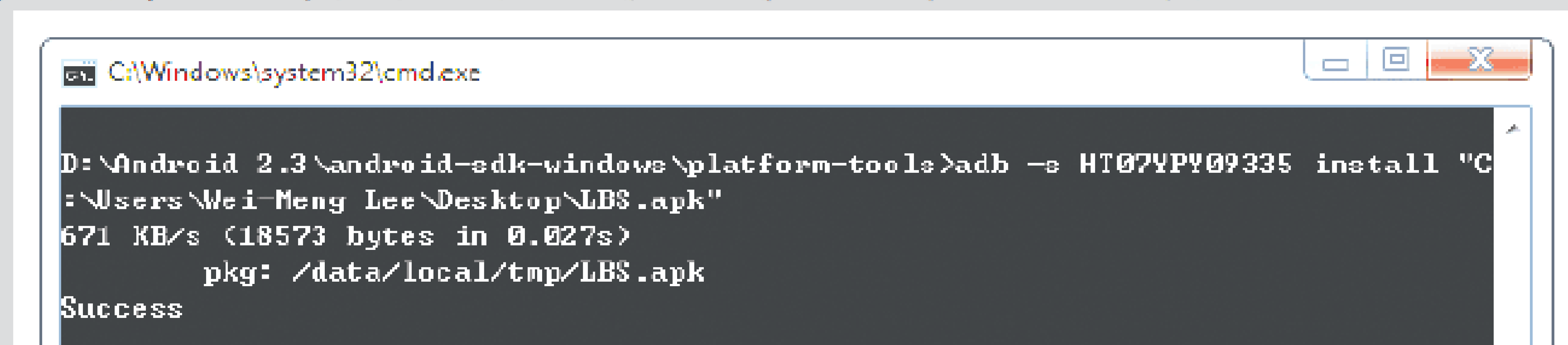


图 11-11

如果查看Android设备/模拟器上的Launcher，可以看到LBS图标(在图11-12的左侧)。如果在Android设备/模拟器上选择Settings | Applications | Manage Applications，将可以看到Where Am I应用程序(在图11-12的右侧)。



图 11-12

adb.exe工具除了可以用来安装应用程序，还可以用来移除应用程序。要做到这一点，可以使用shell选项将一个应用程序从其安装文件夹下移除：

```
adb shell rm /data/app/net.learn2develop.LBS.apk
```

部署应用程序的另外一种方法是使用Eclipse中的DDMS工具(如图11-13所示)。选择一个模拟器(或设备)后，使用DDMS中的File Explorer进入到/data/app文件夹并通过Push a file onto the device按钮将APK文件复制到设备上。

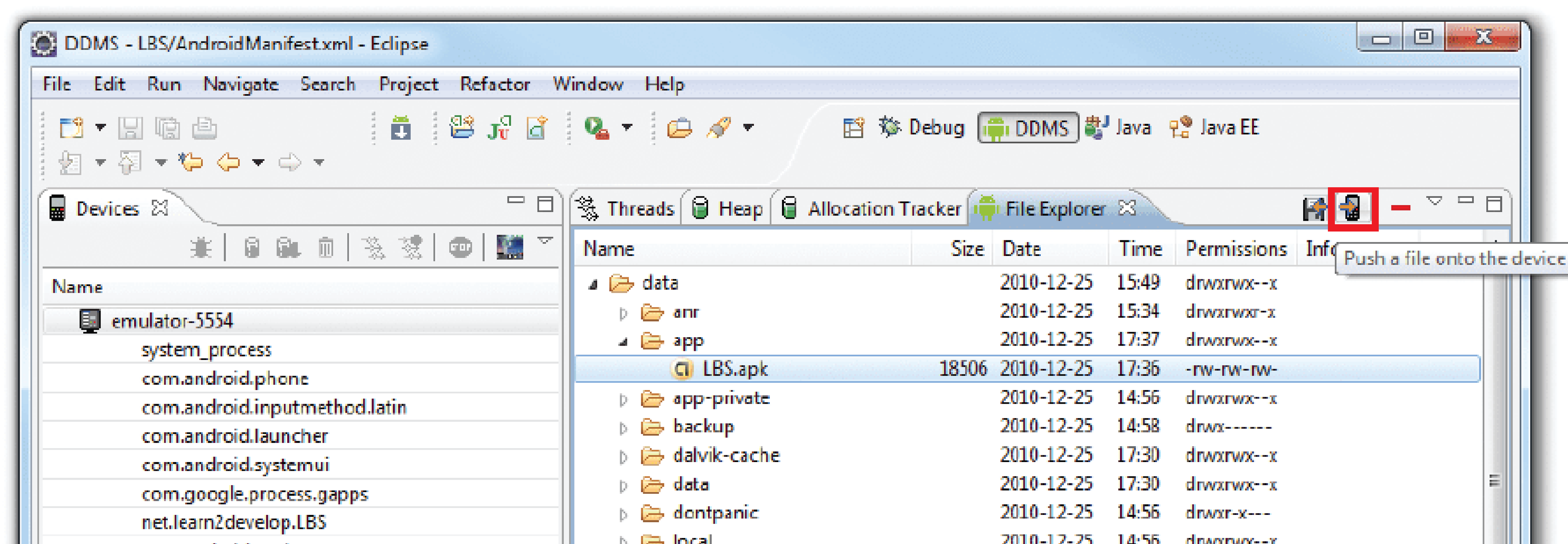
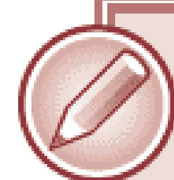


图 11-13

11.2.2　使用Web服务器

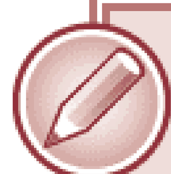
如果希望自己驻留应用程序，可以使用一台Web服务器来完成。如果您有自己的Web托管服务并且打算为您的用户免费提供应用程序(或者将访问限制为特定用户群)，这是个不错的选择。



注意：在下载了APK文件后，即使您将对应用程序的访问限制为某个特定的用户群，也还是没有什么能阻止这些用户将您的应用程序重新分发给其他用户。

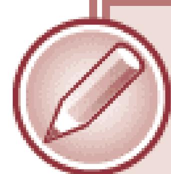
为了说明这一点，我将使用我的Windows 7计算机上的IIS(Internet Information Server)，将已签名的LBS.apk文件复制到c:\inetpub\wwwroot\。另外，创建一个新的名为Install.html的HTML文件，其内容如下：

```
<html>
<title>Where Am I application</title>
<body>
Download the Where Am I application <a href="LBS.apk">here</a>
</body>
</html>
```



注意：如果不清楚如何在Windows 7计算机上设置IIS，可查阅以下链接：http://technet.microsoft.com/en-us/library/cc725762.aspx。

在您的Web服务器上，需要为APK文件注册一个新的MIME类型。.apk扩展名对应的MIME类型是application/vnd.android.package-archive。



注意：要在Web上安装APK文件，需要在您的模拟器或设备上安装一个SD卡。这是由于下载的APK文件将保存在SD卡的download文件夹下。

默认情况下，对于在线安装Android应用程序，Android模拟器或设备只允许安装来自Android Market(www.android.com/market/)的应用程序。因此，对于在Web服务器上的安装，需要配置您的Android模拟器/设备以接受来自非Android Market源的应用程序。

在Application settings菜单下，选中Unknown sources项(如图11-14所示)，会出现一个警告消息，单击OK按钮。选中这一项将允许模拟器/设备可以安装来自非Android Market源的应用程序(例如来自一台Web服务器)。

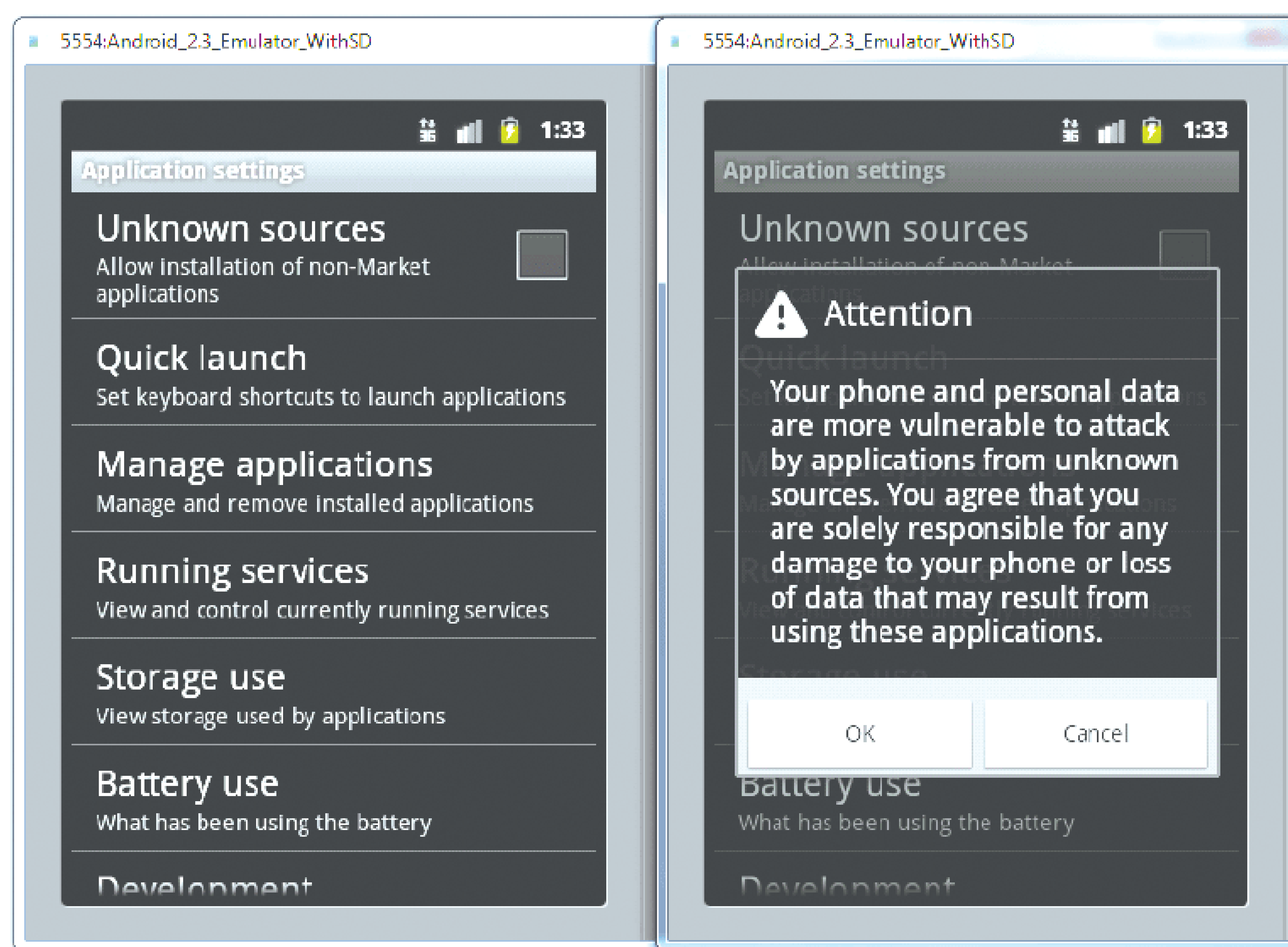


图 11-14

为了从运行在您计算机上的IIS Web服务器安装LBS.apk应用程序，可在Android模拟器/设备上启动Browser应用程序，并导航到指向APK文件的URL。为了指向运行模拟器的计算机，应该使用特殊的IP地址：10.0.2.2。或者，也可以使用主机的IP地址。图11-15展示了Web浏览器上加载的Install.html文件。单击here链接将会把APK文件下载到您的设备上。向下拖动通知栏可以显示下载的状态。

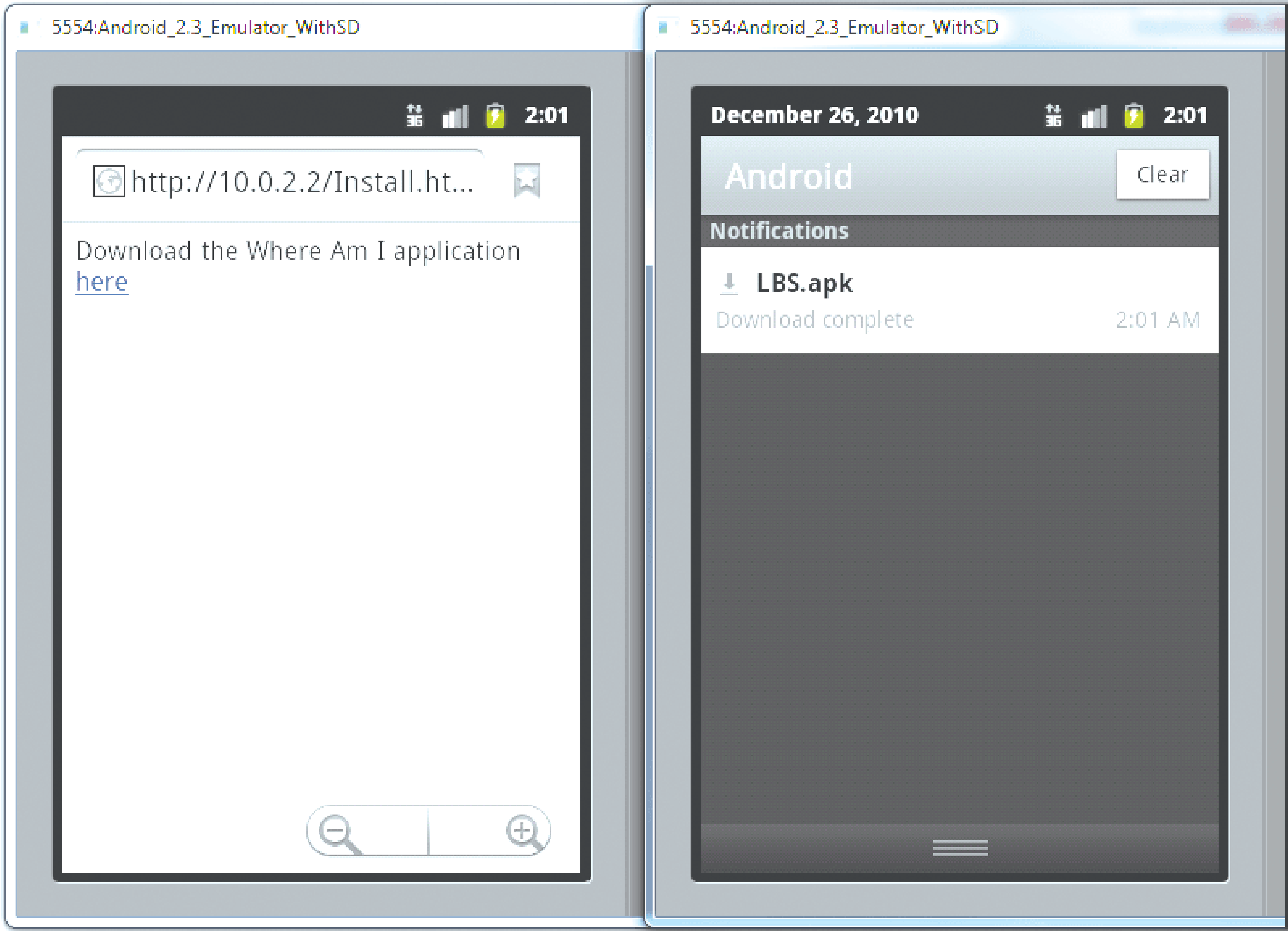


图 11-15

为了安装下载的应用程序，只要轻轻地单击应用程序，将显示出该应用程序所需要的权限(如图11-16所示)。

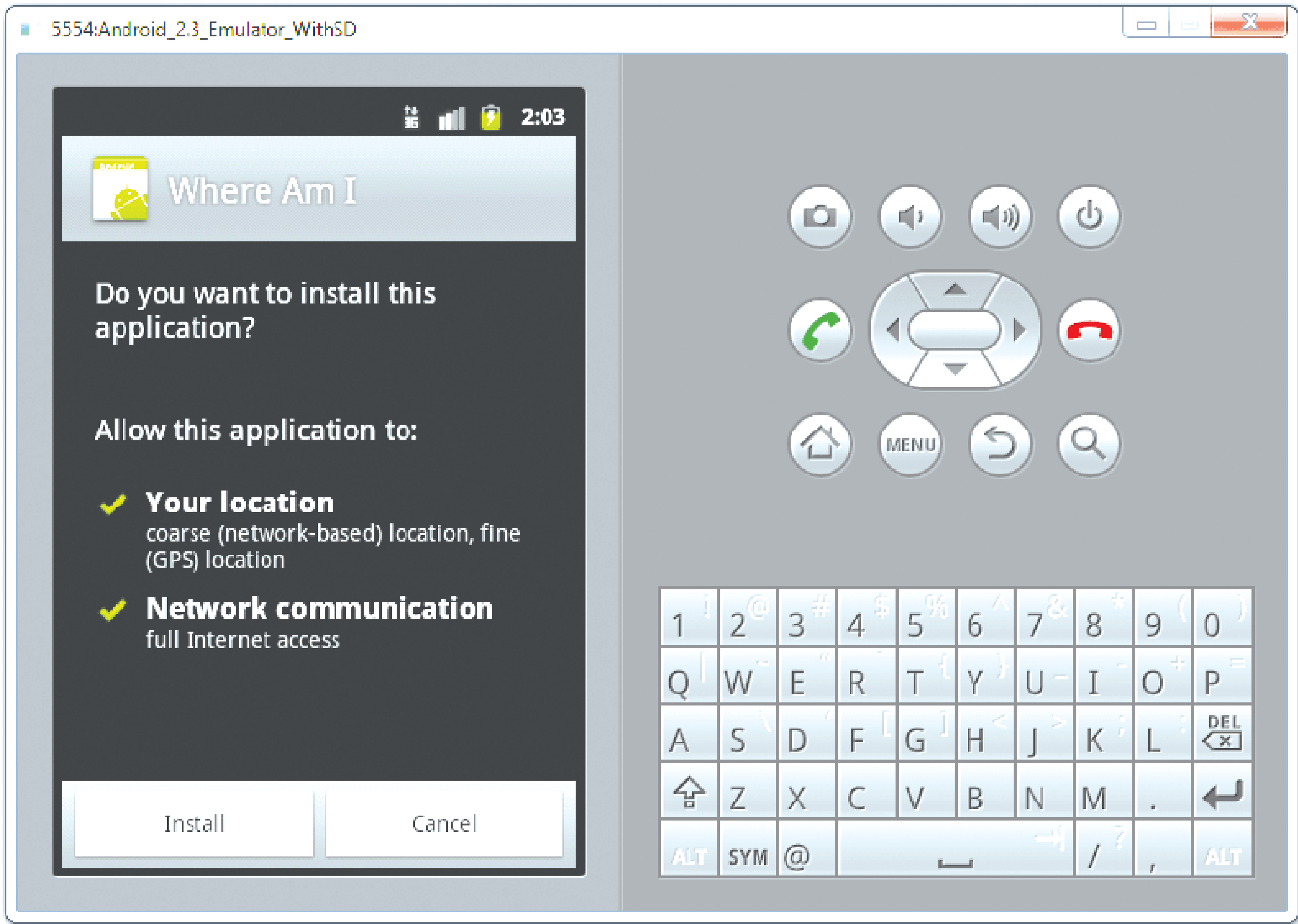


图 11-16

单击Install按钮继续安装。当应用程序安装完毕，可以单击Open按钮来启动它(如图11-17所示)。

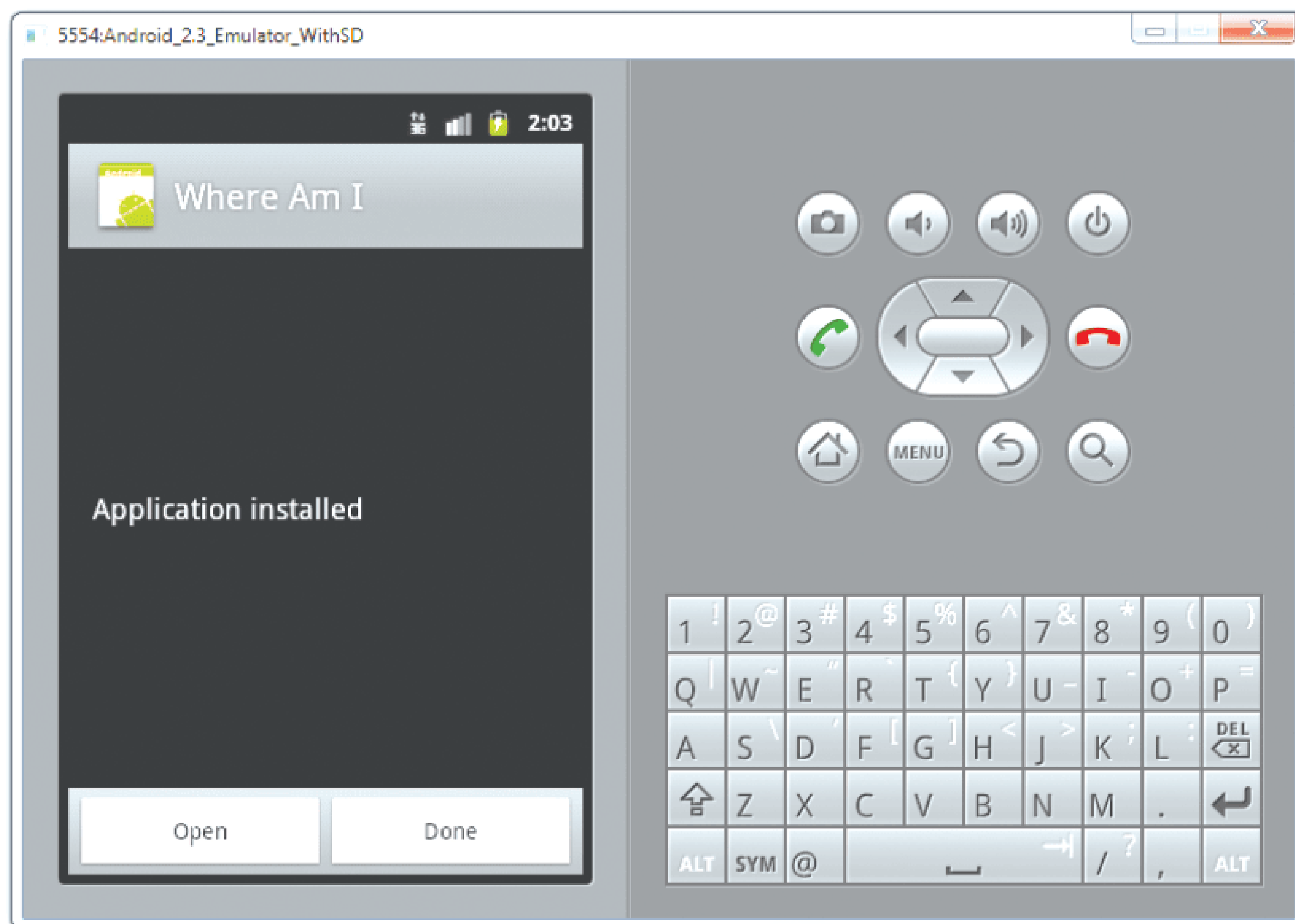


图 11-17

除了使用Web服务器，还可以用电子邮件将应用程序作为附件发给用户。当用户收到邮件时，他们可以下载附件并在自己的设备上直接安装。

11.2.3 在Android Market上发布

到目前为止，您已经学习了如何打包Android应用程序以及分发应用程序所能使用的多种方法——通过Web服务器、adb.exe文件、电子邮件、SD卡等。

然而，这些方法不能为用户提供一个可以很容易地发现您的应用程序的途径。更好的方法是由Android Market托管您的应用程序，这是一个可以使用户为其Android设备发现和下载(购买)应用程序变得非常容易的由Google托管的服务。用户只需要在他们的Android设备上启动Market应用程序，就能够发现可以在他们设备上进行安装的各种应用程序。

在本节中，将学习到如何在Android Market上发布Android应用程序。我将一步步地引导您完成发布，包括在提交给Android Market前应用程序所需要做的各项准备。

1. 创建一个开发者简介

在Android Market上发布应用程序的第一步是在http://market.android.com/publish/Home上创建一个开发者简介。这需要一个Google账户(例如您的Gmail账户)。一旦登录进Android Market，首先创建开发者简介(如图11-18所示)。在输入所要求的信息后单击Continue。

为了在Android Market上发布应用程序，需要支付一次性的注册费用，目前是25美元。单击Google Checkout按钮(如图11-19所示)重定向到一个可以支付注册费用的页面。付费之后，单击Continue链接。

图 11-18

图 11-19

接下来，需要同意Android Market Developer Distribution Agreement。选中I agree复选框并单击I agree. Continue链接(如图11-20所示)。

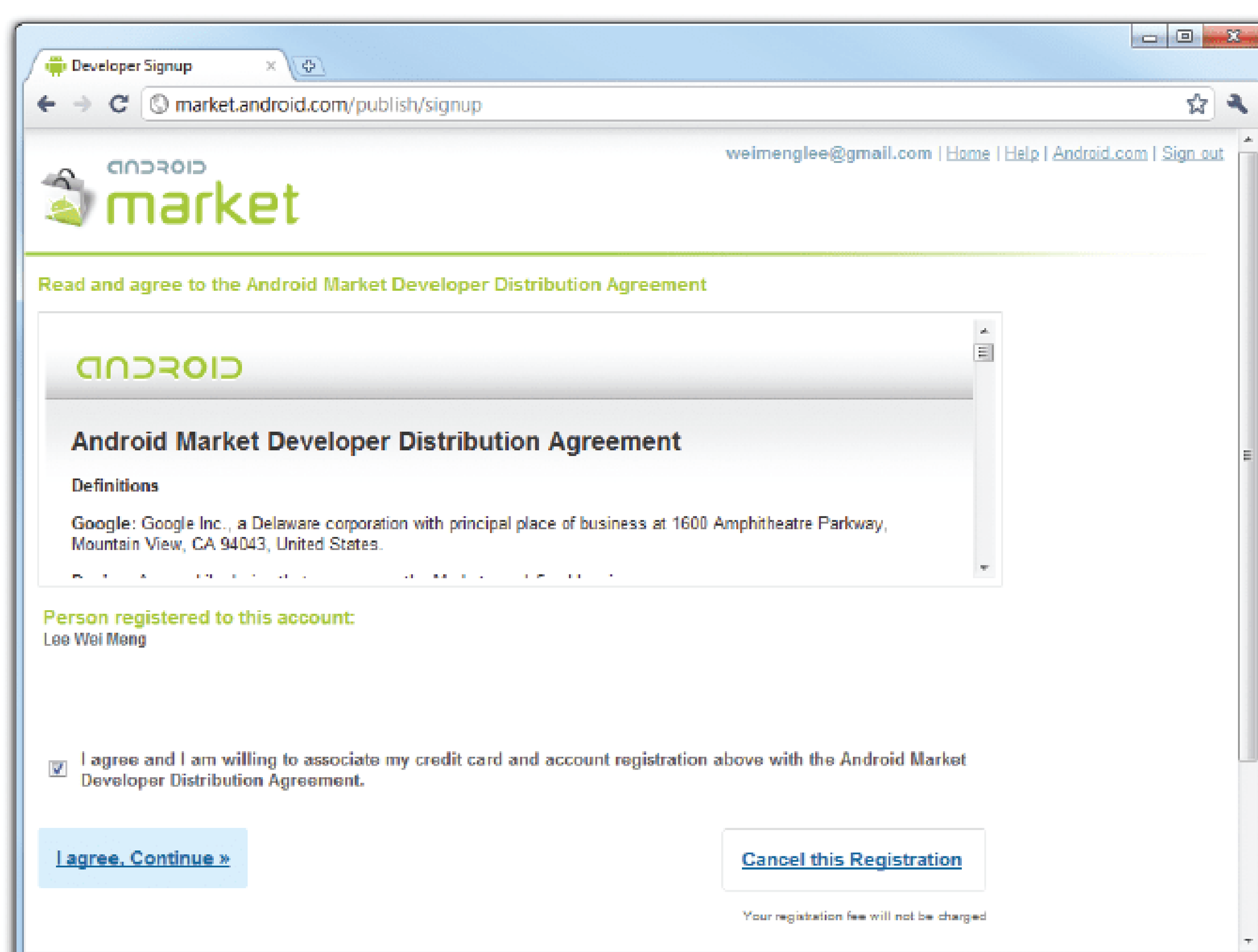


图 11-20

2. 提交应用程序

一旦设置好简介后，就要准备向Android Market提交应用程序了。如果您的应用程序打算收费，单击位于屏幕底部的Setup Merchant Account链接。在这里输入诸如银行账号、税号等额外信息。

对于免费的应用程序，单击Upload Application链接，如图11-21所示。

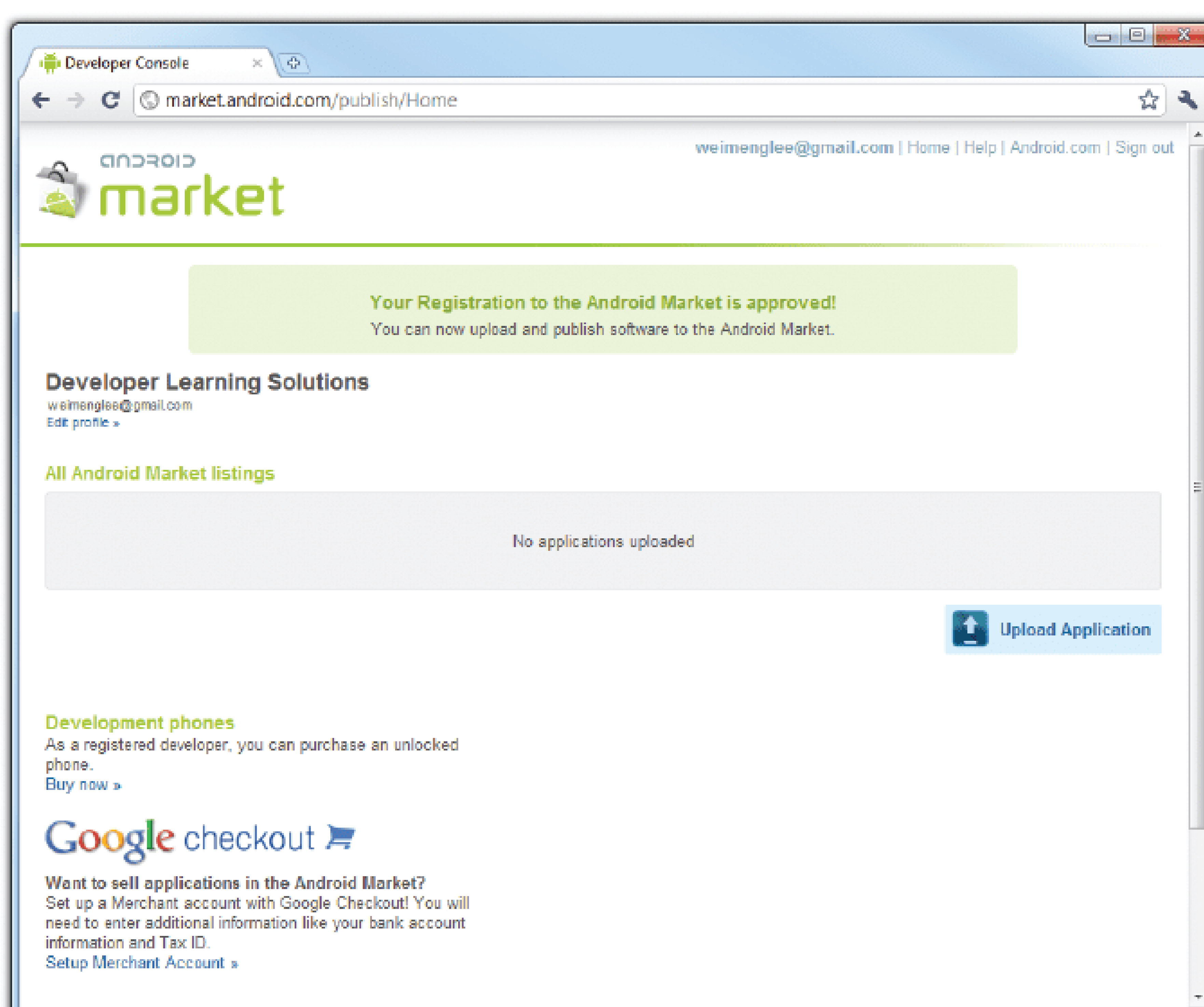


图 11-21

这时，将会要求您输入关于应用程序的一些详细信息。图11-22展示了需要您提供的第一组详细信息。在所需的信息中，必须提供的包括：

- APK格式的应用程序。
- 至少两个屏幕快照。可以使用Eclipse中的DDMS透视图来捕获应用程序在模拟器或真实设备上运行时的快照。

● 一个高分辨率的应用程序图标。图像大小必须是512×512像素。

其他详细信息是可选的，可以以后再提供。

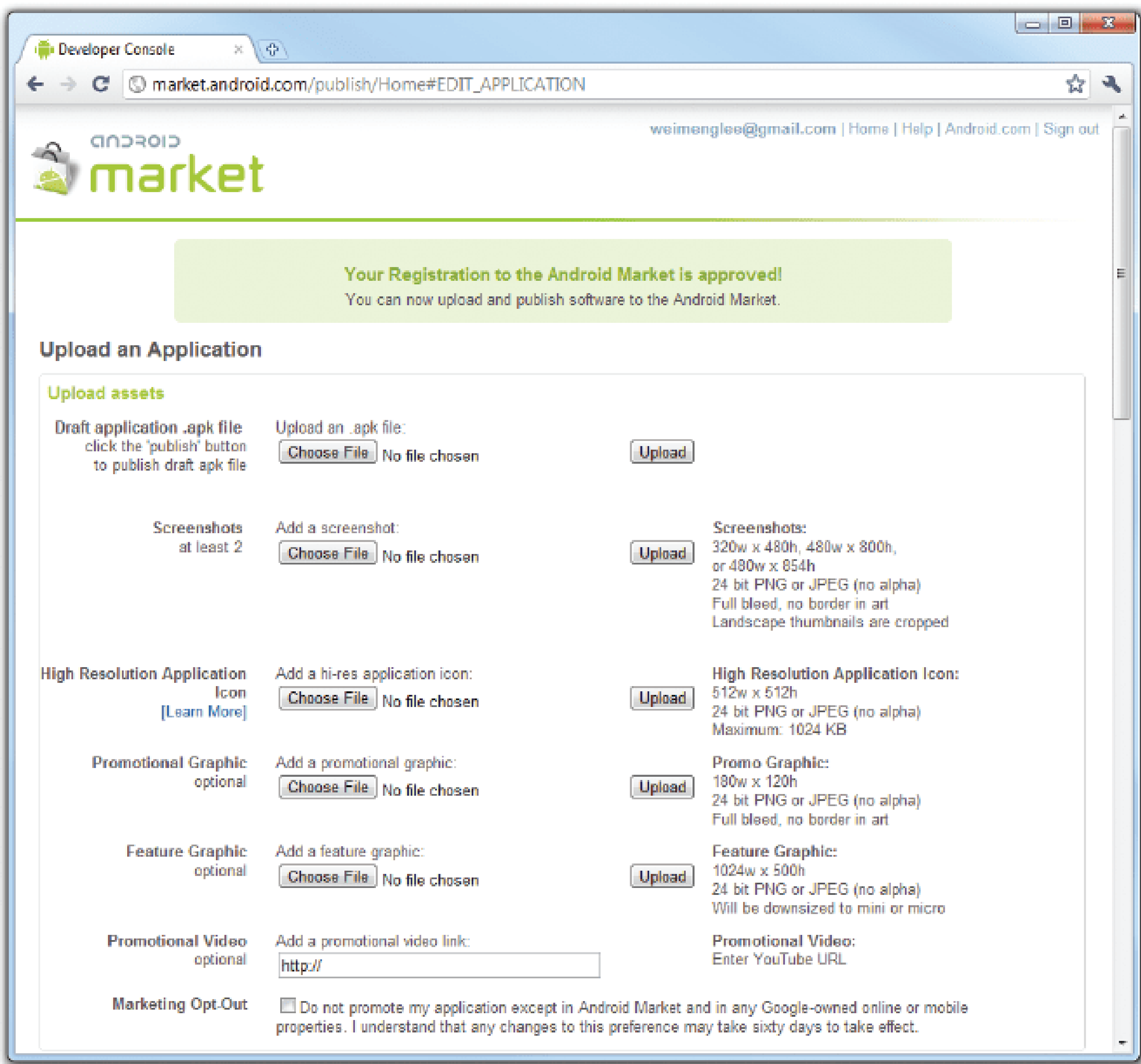


图 11-22

图11-23展示了我已向Android Market站点上传了LBS.apk文件。特别要注意，基于您上传的APK文件，将会有警告消息，告诉用户所需的特定的权限，您的应用程序的功能将被用来作为搜索结果的筛选条件。例如，由于我的应用程序要求接入GPS，对于那些没有GPS接收器的设备的用户，我的应用程序就不会在他们的搜索结果列表上显示。

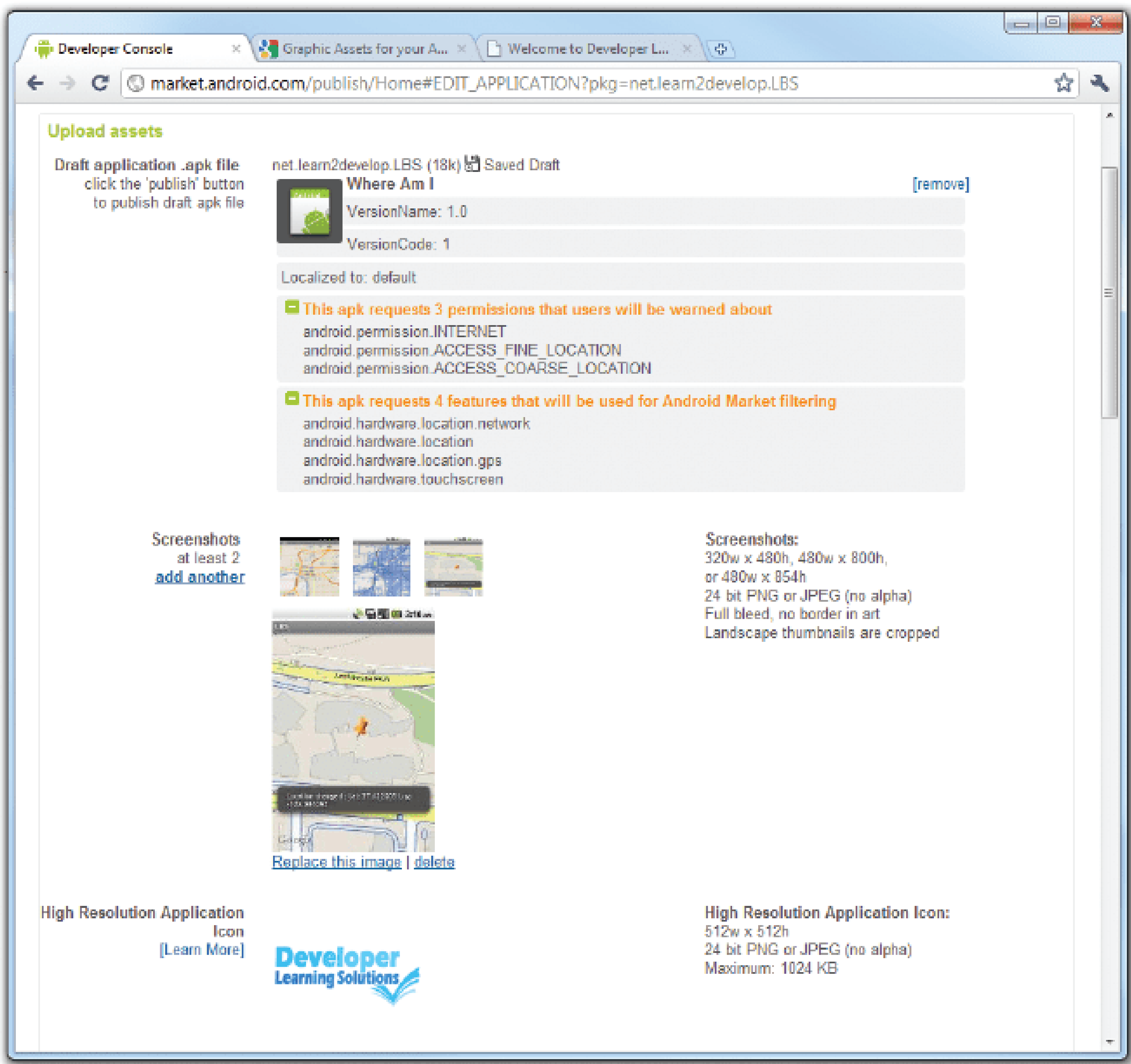


图 11-23

需要提供的下一组信息如图11-24所示，包含了应用程序的标题、其描述以及最新修改的细节(对于应用程序更新很有用)。还可以选择应用程序的类型和在Android Market中所属的类别。

图 11-24

在最后一个对话框中，可以指明您的应用程序是否采用版权保护以及指定内容等级。还可以提供您的网站的URL以及您的联系信息(如图11-25所示)。

图 11-25

当同意了两个准则和协议后，单击Publish按钮将您的应用程序发布到Android Market上。

到此为止，您的应用程序在Android Market上就可用了。您将能够查看任何已提交的关于您的应用程序的评论(如图11-26所示)以及错误报告和总的下载次数。

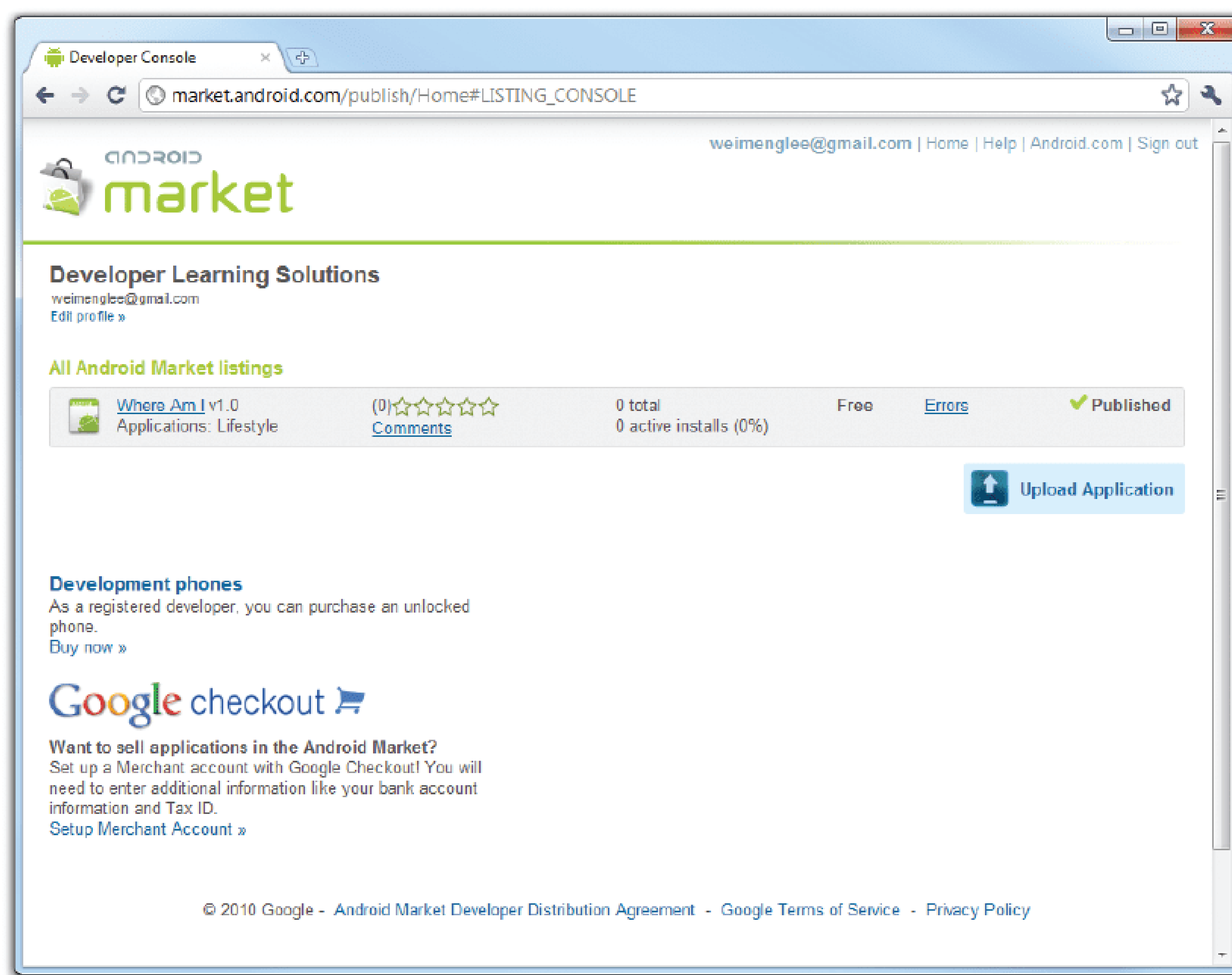


图 11-26

祝您好运！现在，您只要等好消息就行了。希望很快您就能一路笑着去银行！

11.3　本章小结

在本章中，我们学习了如何将Android应用程序导出为一个APK文件并使用自己创建的一个密钥库对其进行数字签名。然后，学习了分发应用程序的多种方法以及每一种方法的优点。最后，我们亲自体验了在Android Market上发布应用程序所需的步骤，这将使您可以出售您的应用程序，并且带来更多的用户。希望这样可以使您卖出更多的产品并因此获得不错的收益！

练　习

1. 如何指定应用程序所需的Android最低版本？
2. 如何生成一个自签名证书来为Android应用程序签名？
3. 如何配置Android设备，使其可以接收非Anroid Market源的应用程序？

练习答案参见附录C。

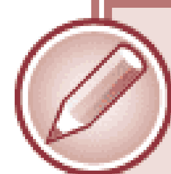
本章主要内容

主　　题	关键概念
用于发布应用程序的检查表	要在Android Market上发布一个应用程序，该应用程序必须在Android-Manifest.xml文件中具有以下4个属性：android:versionCode、android:versionName、android:icon和android:label
应用程序必须签名	要分发的所有应用程序必须使用一个自签名证书进行签名。调试密钥库对于分发来说是无效的
导出一个应用程序并对其签名	使用Eclipse的Export功能将应用程序导出为一个APK文件并使用一个自签名证书对其签名
部署APK文件	可以使用各种方式部署：Web服务器、电子邮件、adb.exe和DDMS等
在Android Market上发布应用程序	花费25美元向Android Market进行申请，这样就可以在Android Market上托管和出售应用程序

附录 A

使用Eclipse进行Android开发

尽管Google支持使用诸如IntelliJ这样的IDE或者像Emacs这样的基本编辑器来进行Android应用程序的开发，但它还是推荐将Eclipse IDE和ADT插件一起使用。这样做可以使开发Android应用程序变得更容易也更高效。本附录描述了Eclipse中可用的一些可以使您的开发工作变得更加容易的极好功能。



注意：如果您还没有下载Eclipse，那么可以从第1章开始学起。在那里，您将学到如何获取Eclipse以及对它进行配置，使其可以和Android SDK一起使用。本附录假定您已经为Android开发设置好了Eclipse环境。

A.1 Eclipse概览

Eclipse是一个高度可扩展的多语言软件开发环境，可以支持各种应用程序开发。使用Eclipse，可以利用多种语言来编写和测试应用程序，例如Java、C、C++、PHP、Ruby等。由于其可扩展性，Eclipse的新手常常感到对其IDE无所适从。因此，以下小节的内容旨在帮助您在Android应用程序的开发过程中可以更加熟练地使用Eclipse。

A.1.1 工作区

Eclipse采用工作区(workspace)的概念。所谓工作区就是您所选择的用来保存所有项目的一个文件夹。

当第一次启动Eclipse时，将提示您选择一个工作区(如图A-1所示)。

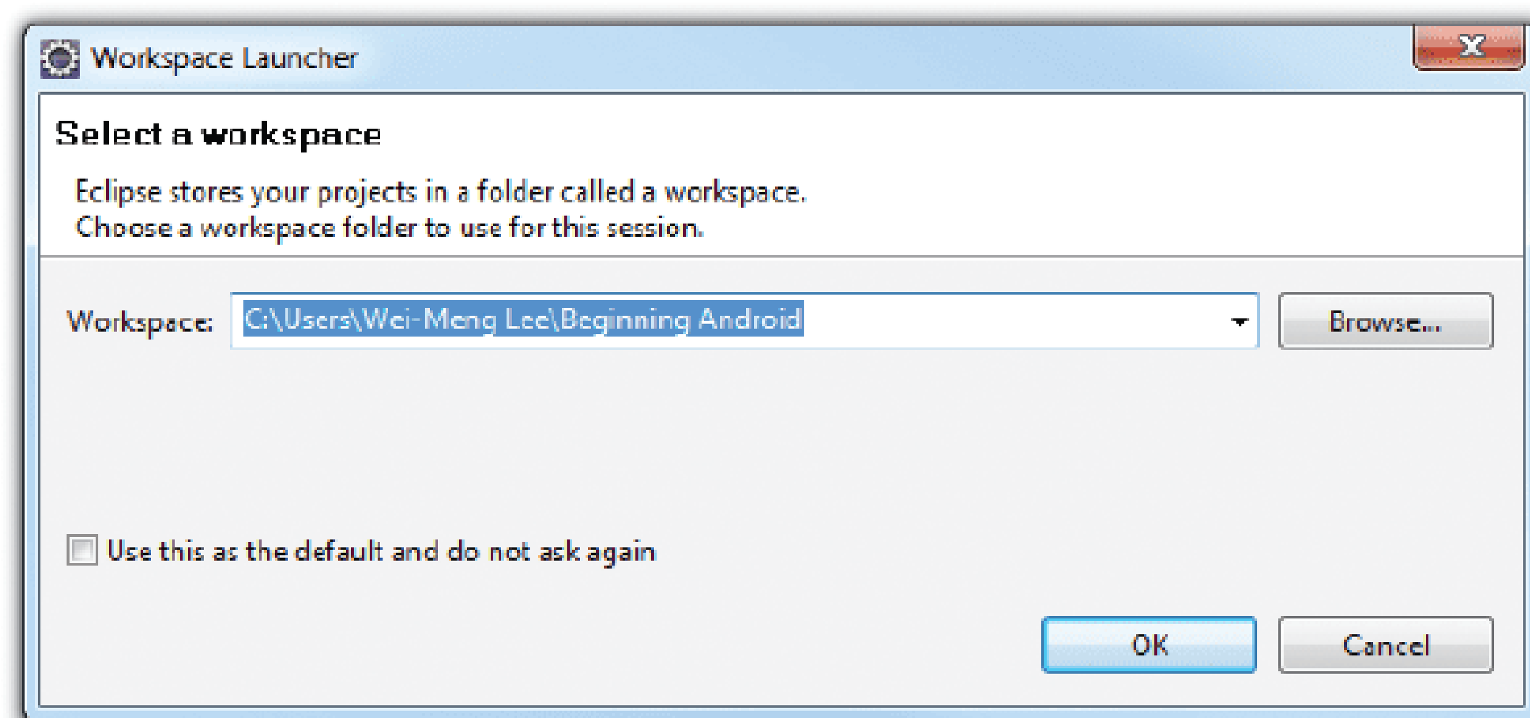


图 A-1

当Eclipse将位于工作区中的项目启动完之后，将在IDE中显示几个窗格(如图A-2所示)。

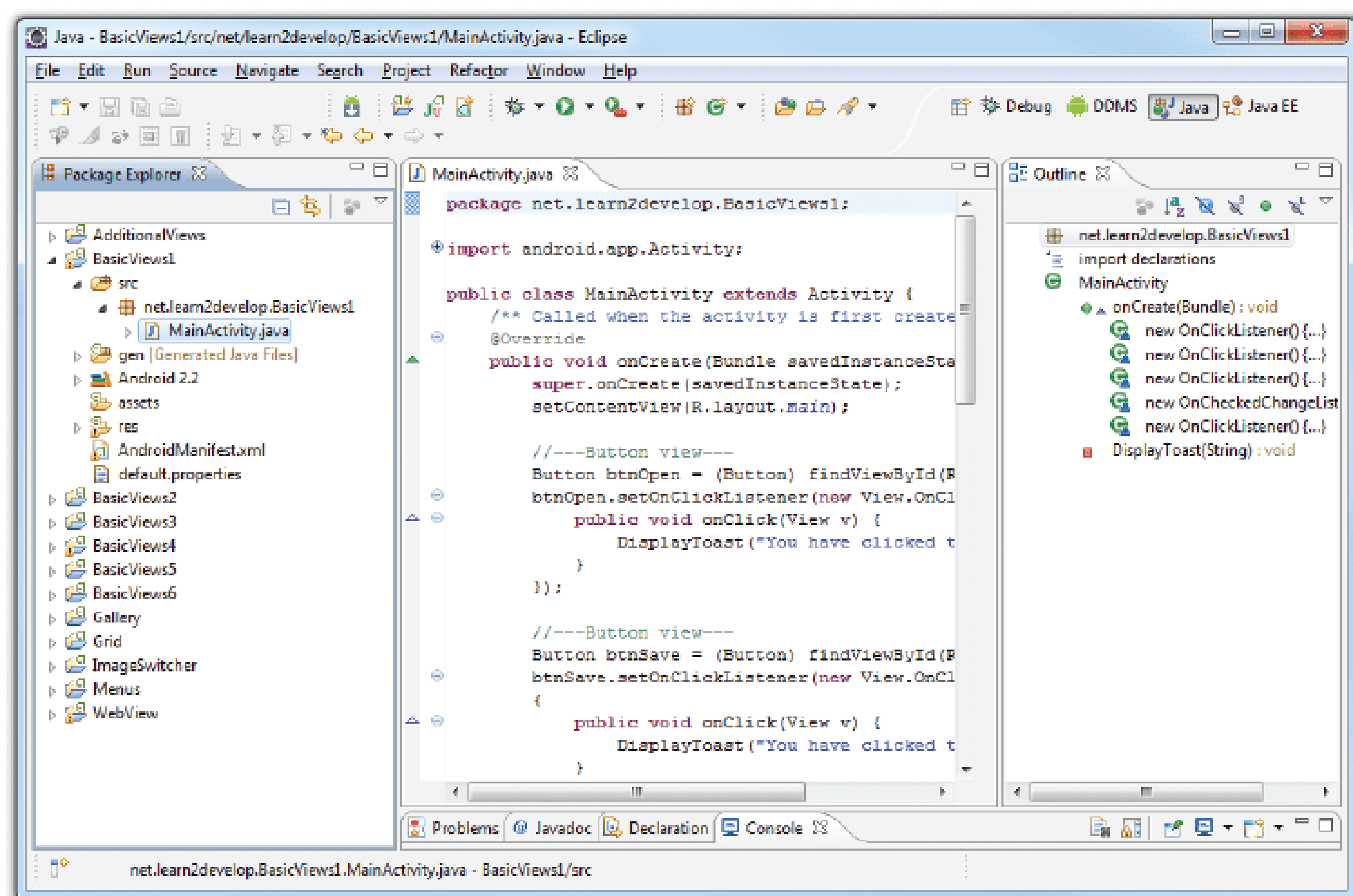


图 A-2

下面的小节将对在开发Android应用程序时必须了解的一些较重要的窗格进行重点讲解。

A.1.2 Package Explorer

如图A-3所示，Package Explorer列出了当前位于工作区中的所有项目。要编辑项目中一个特定的项，可以双击这一项，文件将会显示在各自的编辑器中。

还可以右击在Package Explorer中列出的每一项，以显示与所选项有关的上下文敏感菜单。例如，如果希望在项目中添加一个新的.java文件，可以在Package Explorer中右击包的名称，然后选择New | Class (如图A-4所示)。

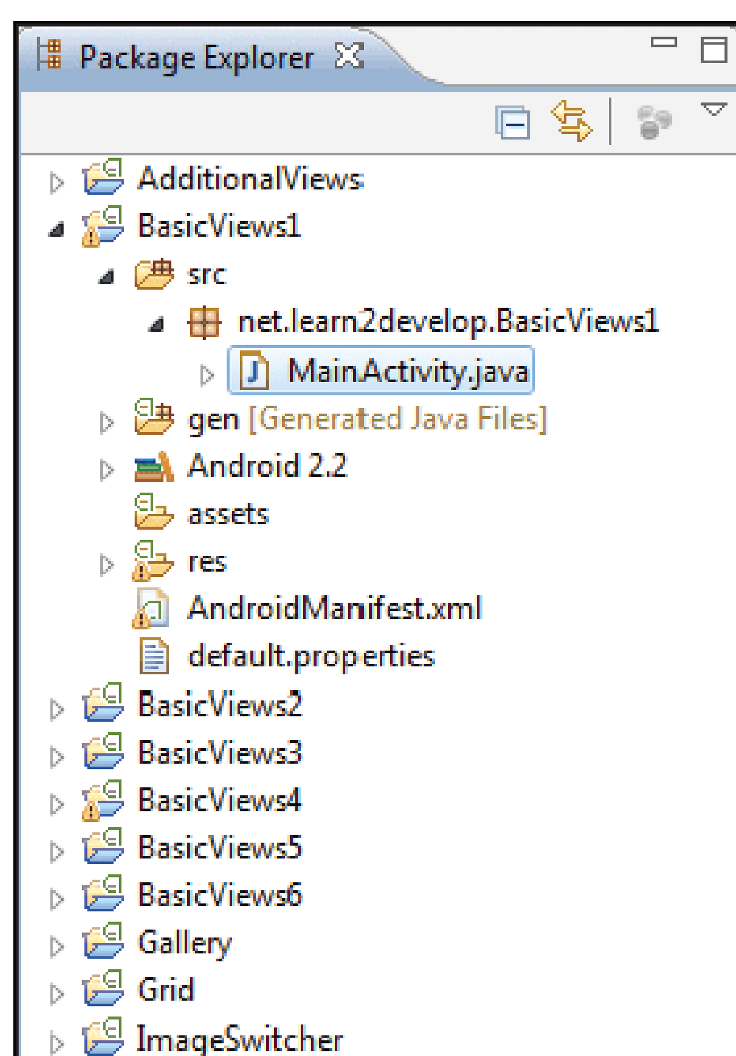


图 A-3

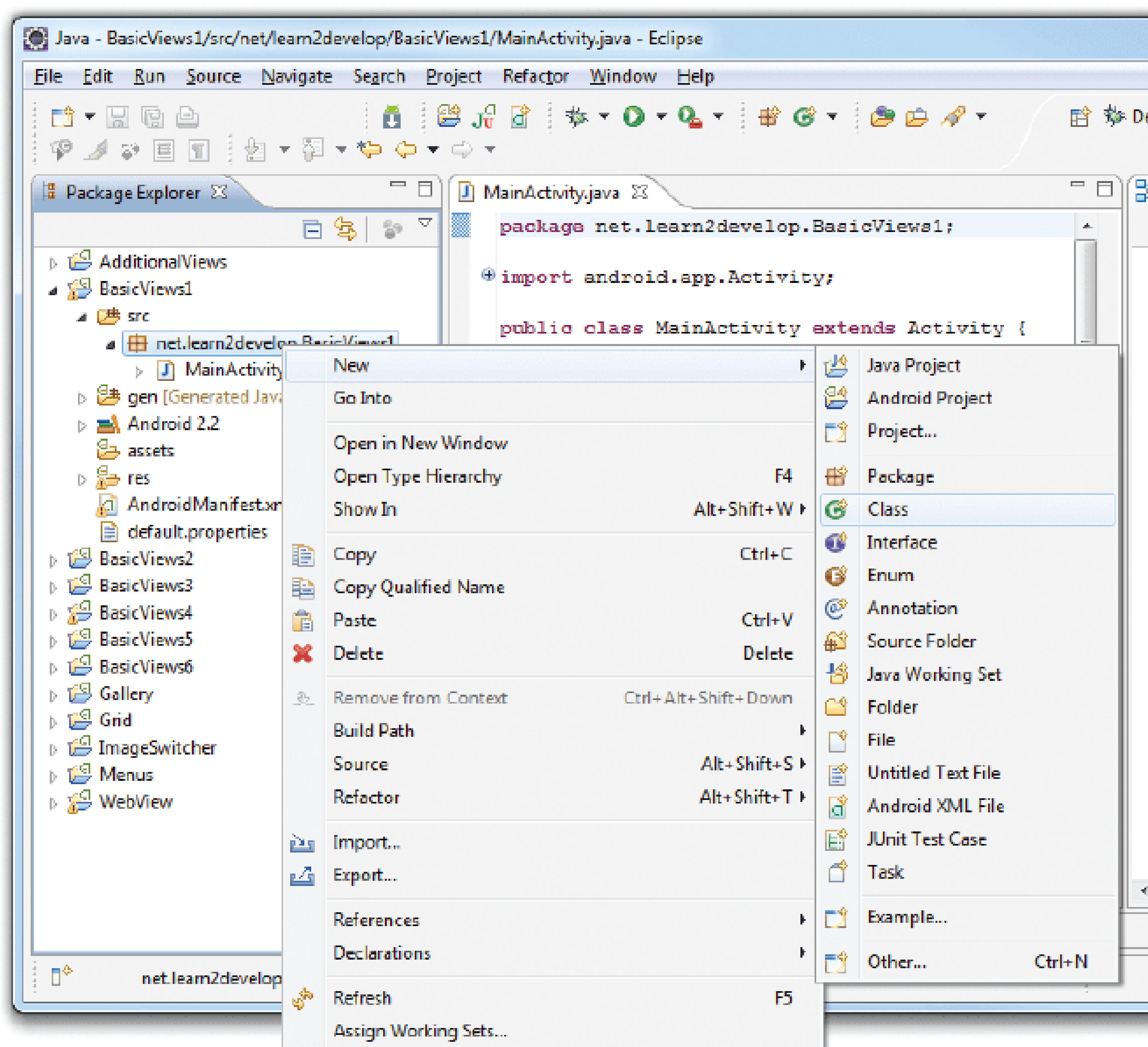


图 A-4

A.1.3　使用其他工作区的项目

也许您有时候会创建多个工作区来存储不同的项目。如果需要访问另一个工作区中的项目，通常有两种方法可以实现这一点。

第一种方法是选择File | Switch Workspace(如图A-5所示)切换到您所希望的工作区。指定新工作区后重新启动Eclipse。

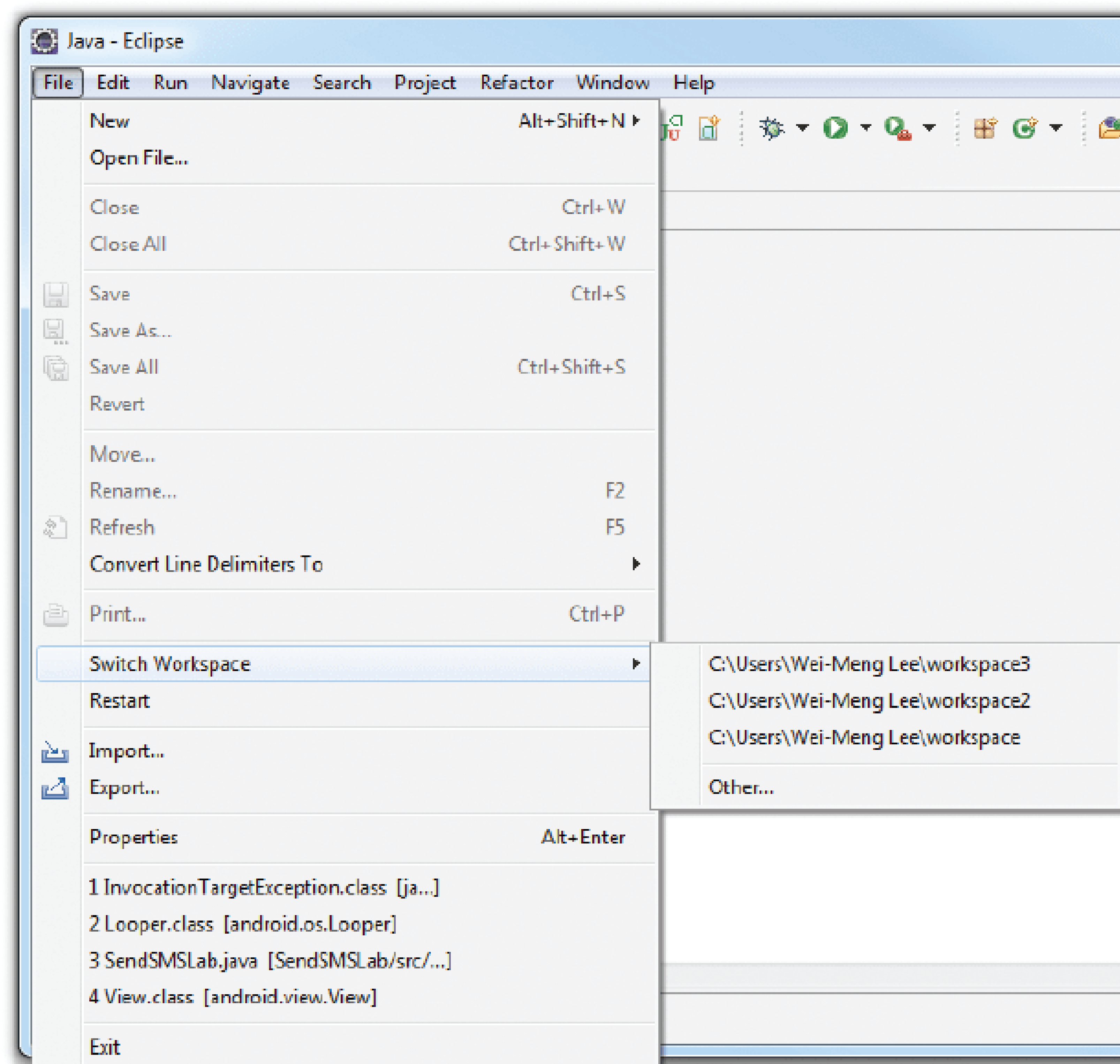


图 A-5

第二种方法是将项目从另一个工作区导入到当前工作区中。要做到这一点，选择File | Import...，然后选择General | Existing Projects into Workspace(如图A-6所示)。

在Select root directory文本框中输入包含想要导入的项目的工作区的路径，并选中这些项目(如图A-7所示)。单击Finish按钮，导入所选择的项目。

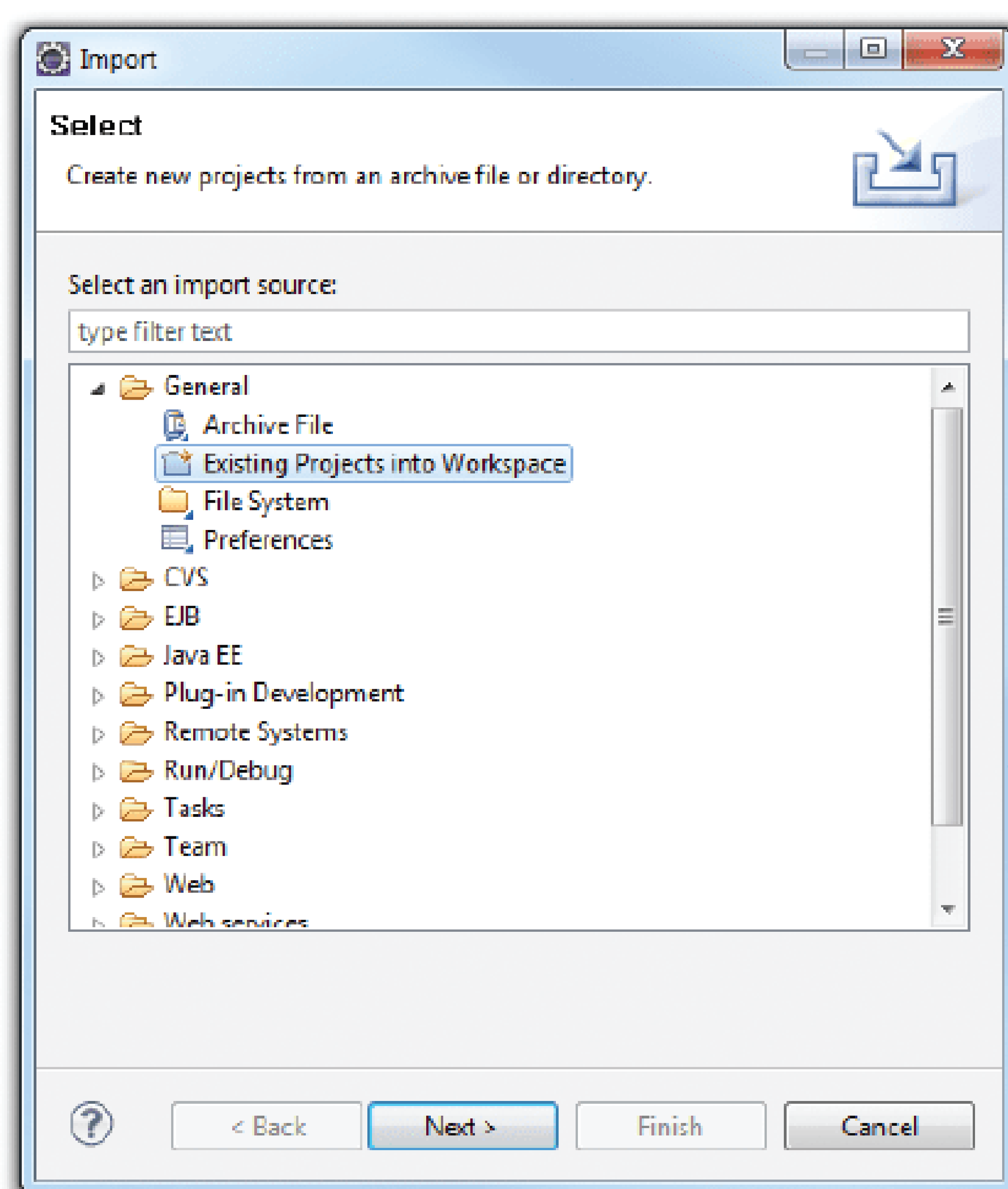


图 A-6

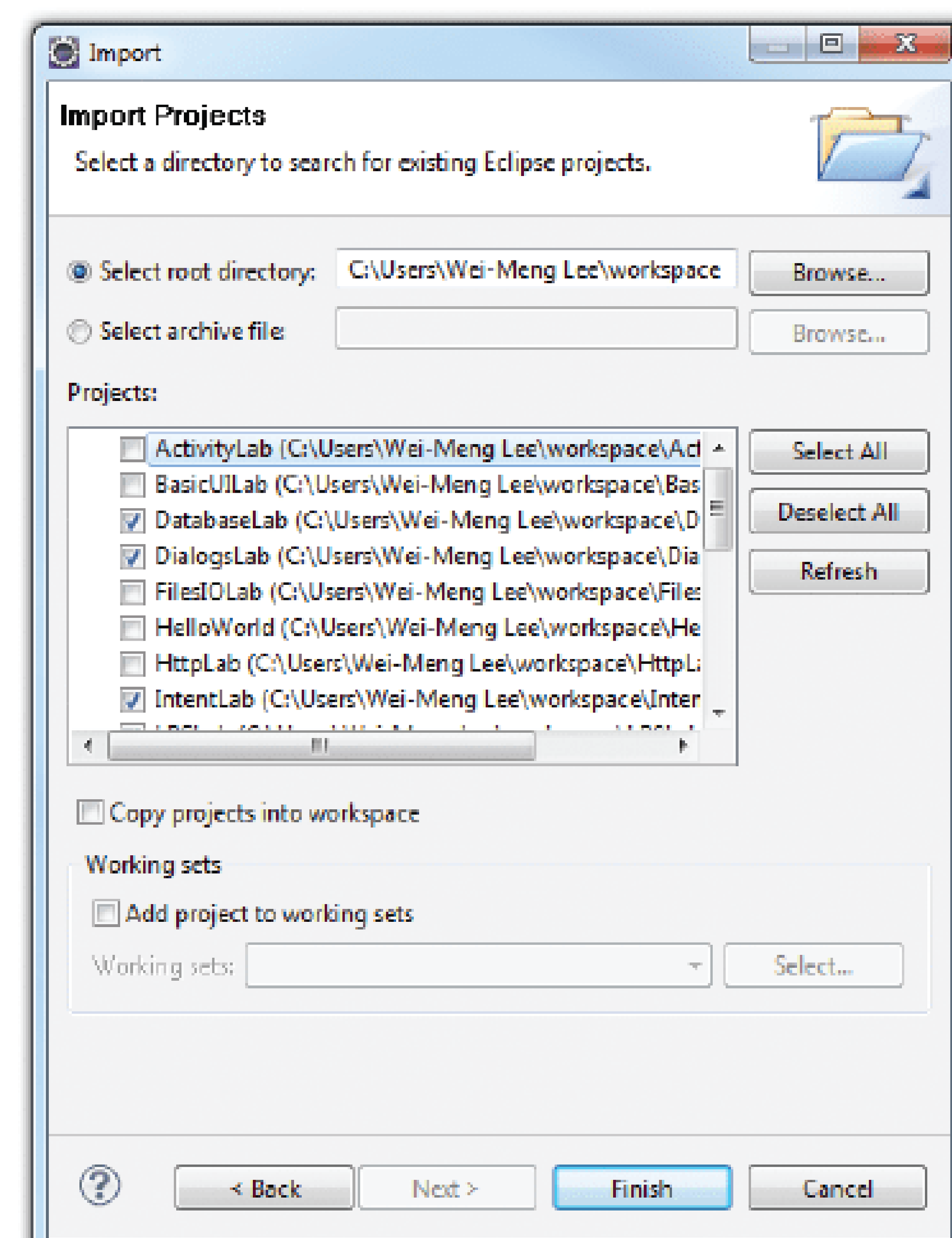


图 A-7

注意，当从另一个工作区将一个项目导入到当前工作区中时，导入项目的物理位置保持不变。也就是说，项目仍旧位于其原始目录下。要在当前工作区中保留项目的一个副本，可选中Copy projects into workspace选项。

A.1.4 编辑器

根据在Package Explorer中双击的项目的类型，Eclipse将为您打开对应的编辑器来编辑文件。举例来说，如果双击一个.java文件，将打开用于编辑源文件的文本编辑器(如图A-8所示)。

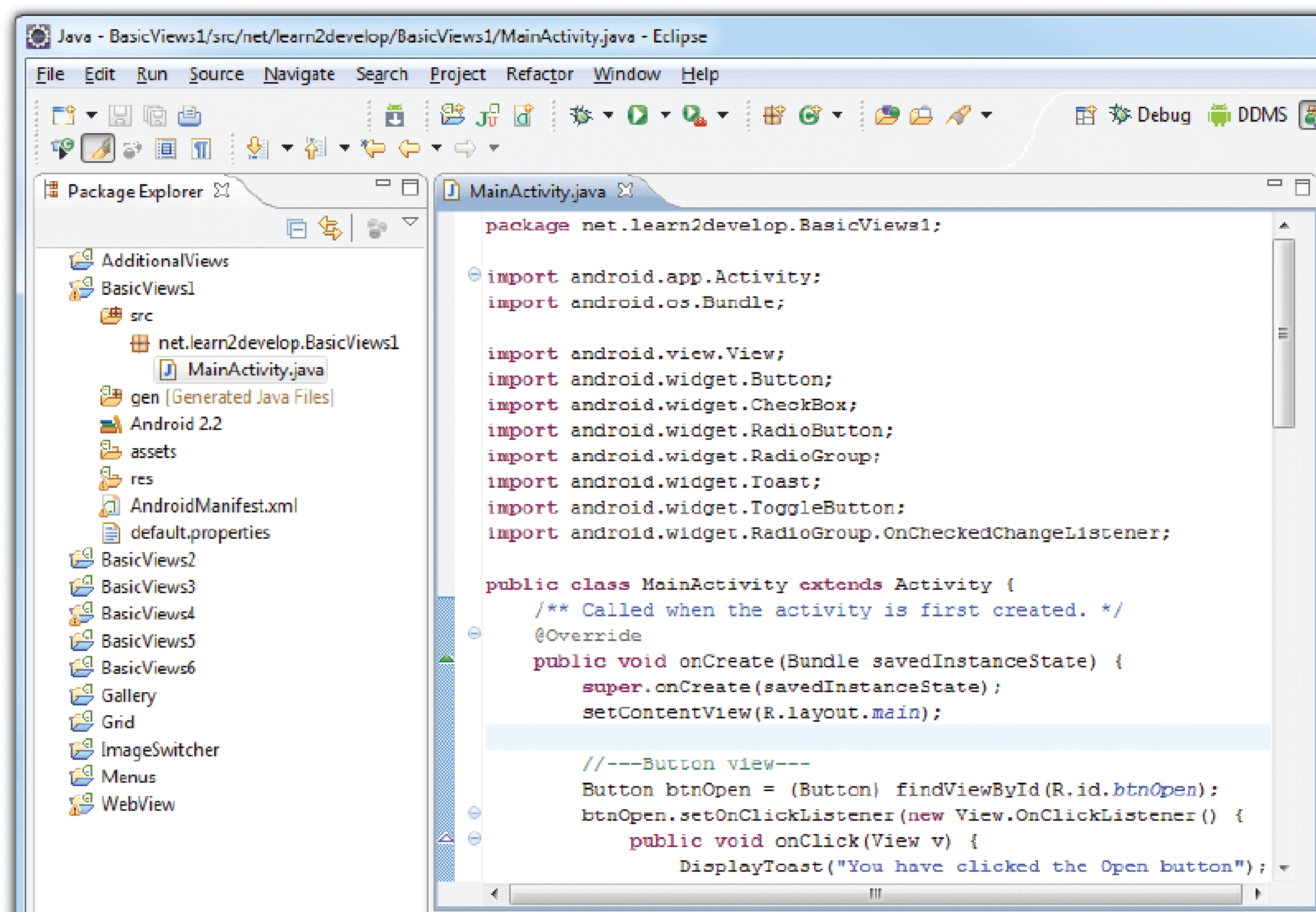


图 A-8

如果双击res/drawable-mdpi文件夹下的icon.png文件，将启动Windows Photo Viewer应用程序来显示图像(如图A-9所示)。

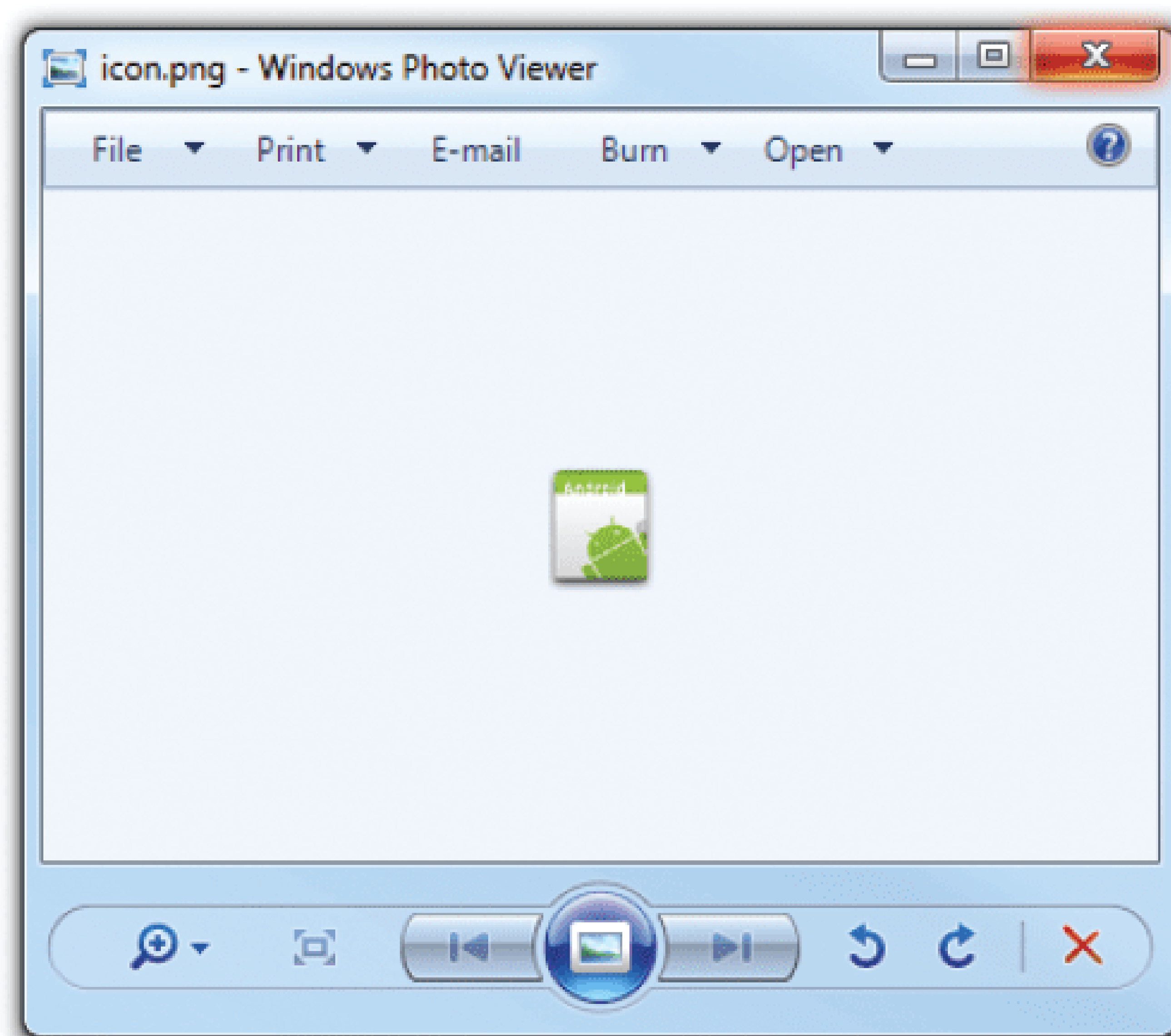


图 A-9

如果双击res/layout文件夹下的main.xml文件，Eclipse将显示UI编辑器，在那里可以以图形化方式查看和构建UI布局(如图A-10所示)。

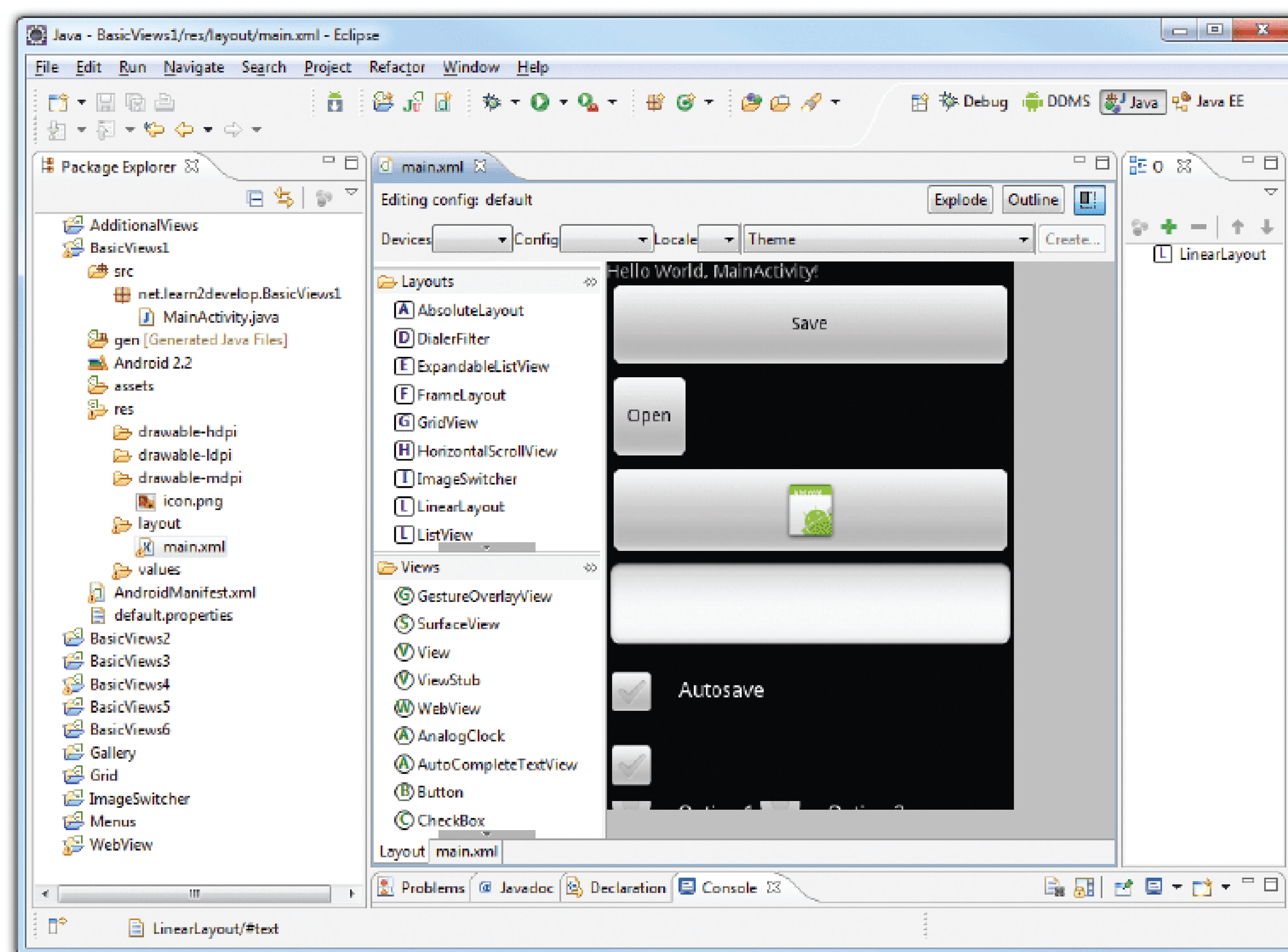


图 A-10

要使用XML手动编辑UI，可以单击位于屏幕底部的main.xml选项卡，切换到XML视图(如图A-11所示)。

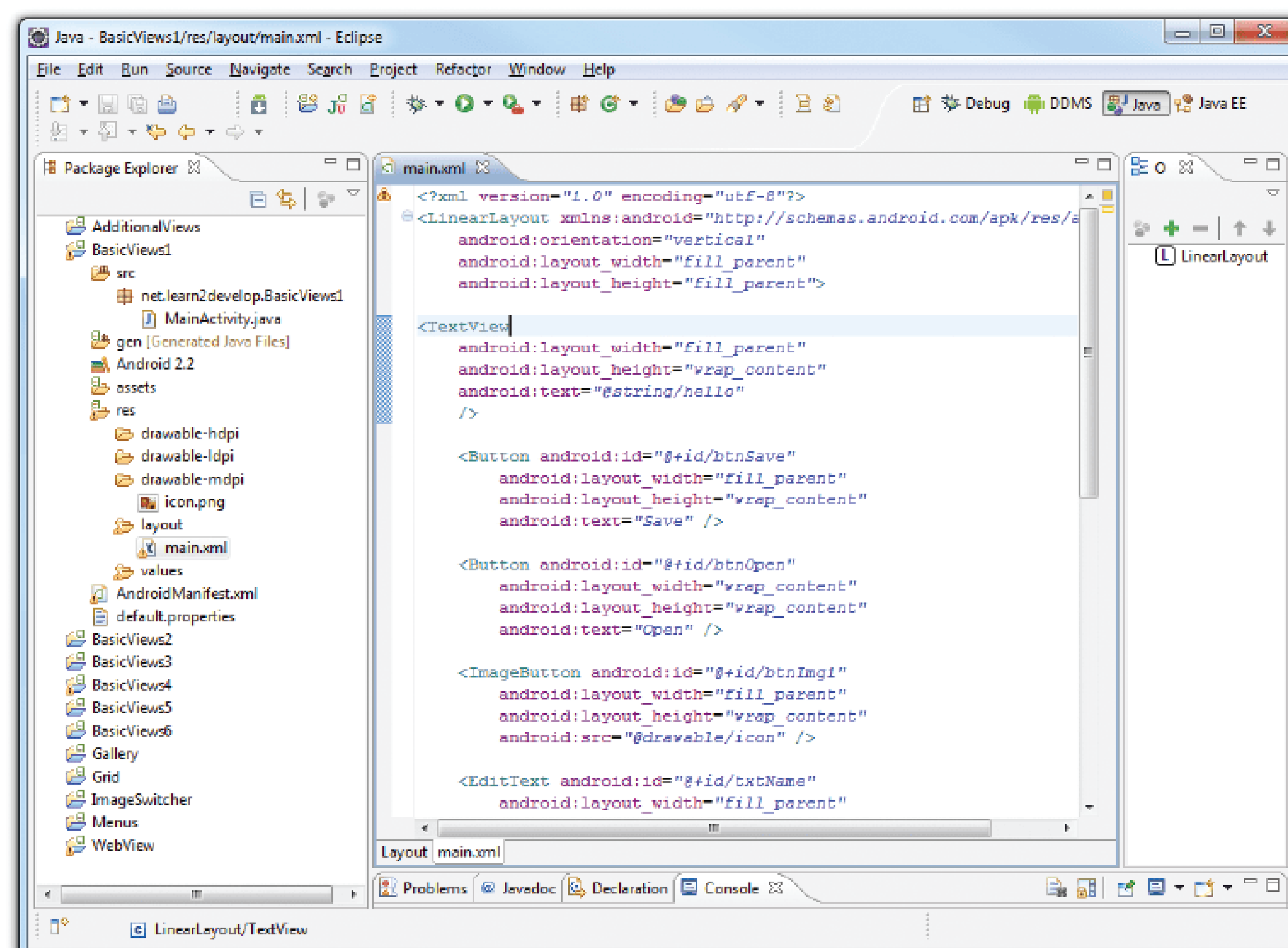


图 A-11

A.1.5　透视图

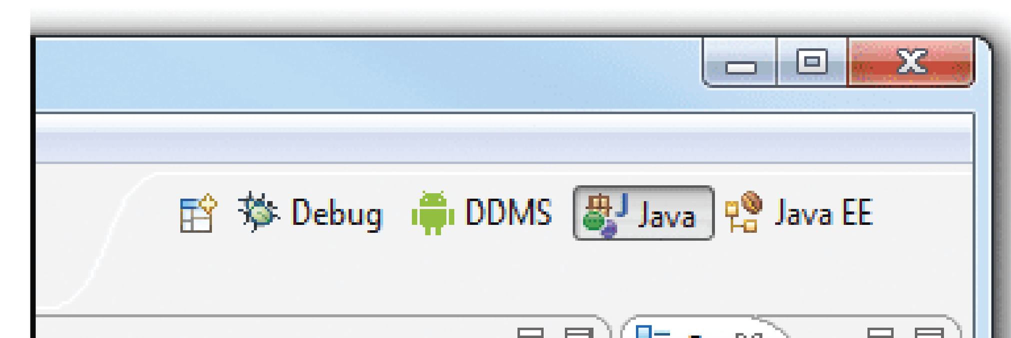


图 A-12

在Eclipse中，透视图(perspective)是一个包含一组视图和编辑器的可视化容器。当在Eclipse中编辑项目时，您就处于Java透视图中(如图A-12所示)。

Java EE透视图用于开发企业级的Java应用程序，它包含

了与之相关的其他模块。

通过单击透视图的名称可以进行透视图的切换。如果透视图名称没有显示，可以单击Open Perspective按钮添加一个新的透视图(如图A-13所示)。

DDMS透视图包含了与Android模拟器和设备进行通信的工具。在附录B中将进行详细介绍。Debug透视图包含了用于调试Android应用程序的窗格，本附录稍后将对此进行详细介绍。

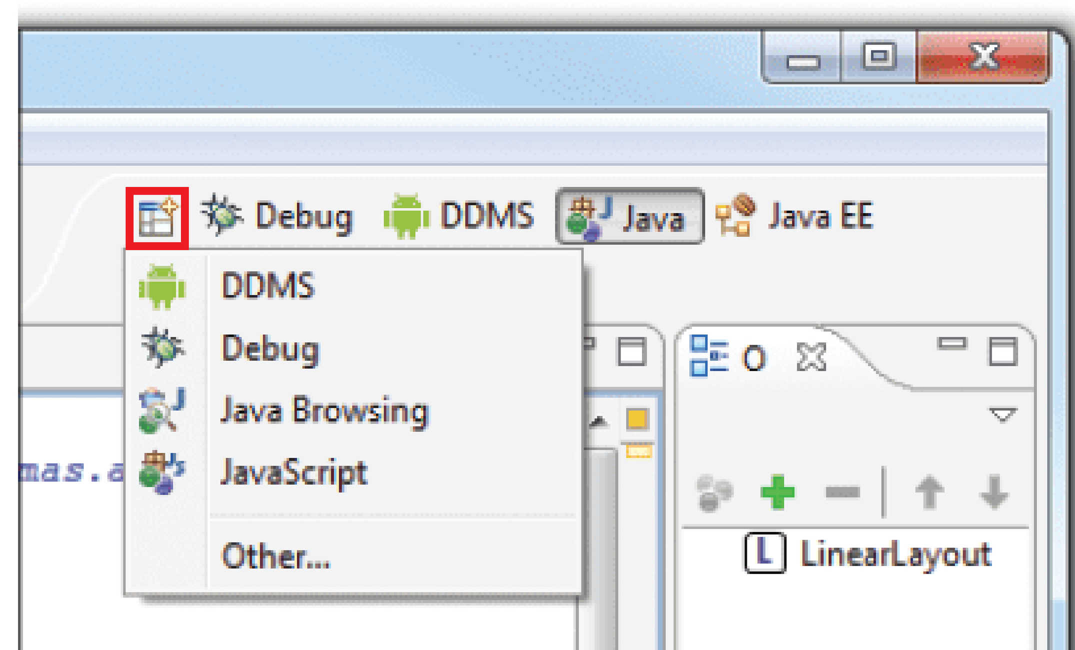


图 A-13

A.1.6 命名空间的自动导入

Android库中的各种类是以命名空间的形式组织起来的。因此，当从一个命名空间使用一个特定的类时，需要导入适当的命名空间，如下所示：

```
import android.app.Activity;
import android.os.Bundle;
```

由于Android库中类的数目非常巨大，要记住每一个类所属的正确的命名空间不是一件容易的事。幸运的是，Eclipse可以帮您找到正确的命名空间，使得您只要单击鼠标就可以导入它。

图A-14展示了我所声明的一个Button类型的对象。由于我没有为Button类导入正确的命名空间，Eclipse在语句下面提示一个错误。当将鼠标移动到Button类的上面时，Eclipse会显示一个修改建议列表。在本例中，我需要导入android.widget.Button命名空间。单击Import ‘Button’(android.widget)链接将在文件开头添加导入语句。或者，可以使用如下的组合键：Control+Shift+o。这一组合键将使Eclipse自动导入您的类所需要的所有命名空间。

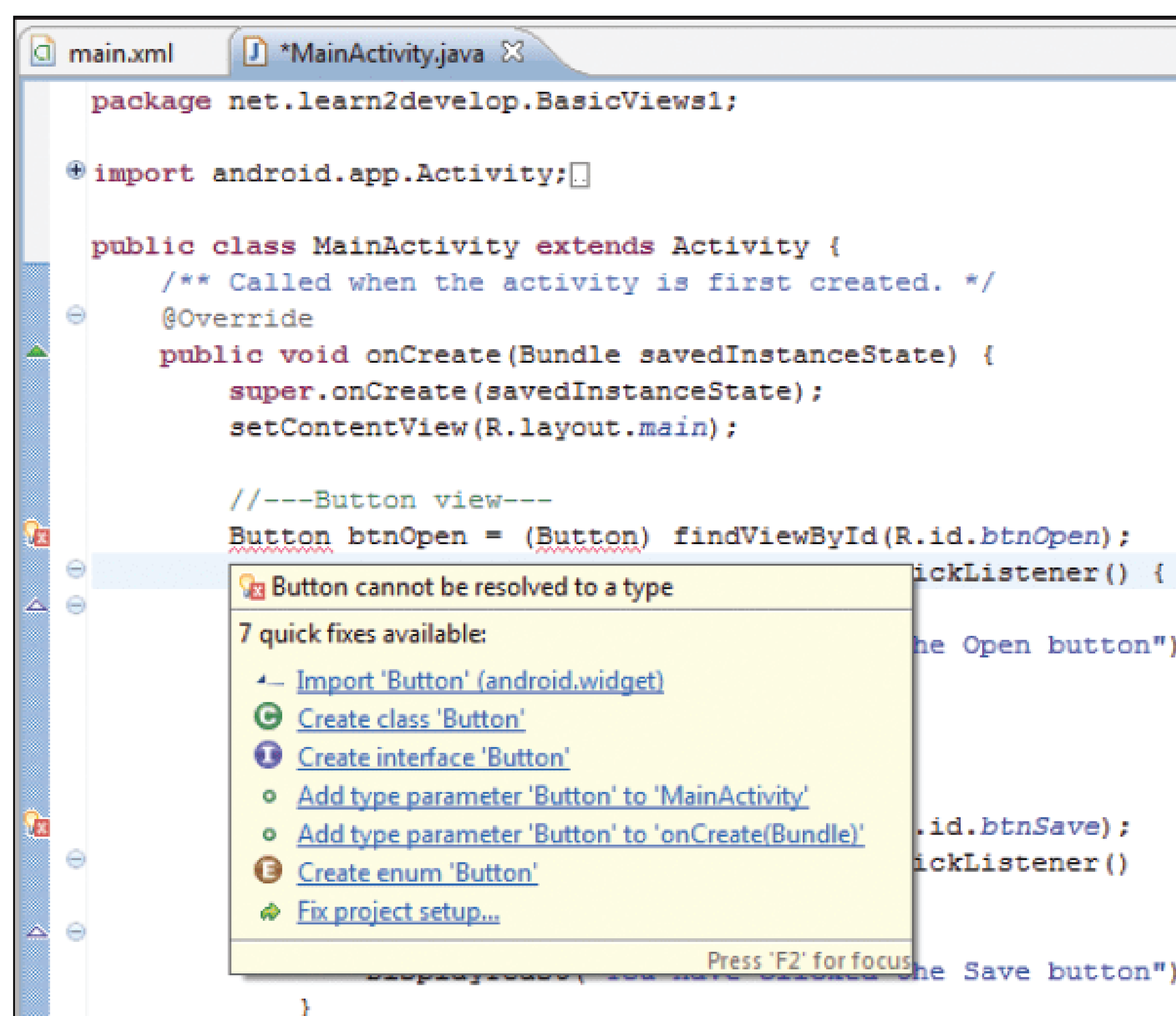


图 A-14

A.1.7 代码完成

Eclipse另一个非常有用的功能就是支持代码完成。当您在代码编辑器中输入内容时，代码完成会显示一个上下文敏感的相关类、对象、方法以及属性名称的列表。例如，图A-15展示了起作用的代码完成功能。在我输入单词fin时，通过按下Ctrl+Space组合键能够激活代码完成功能，这时将出现一个以fin开头的名称列表。

要选择所需的名称，直接在其上双击或者使用光标高亮显示它并按Enter键就行了。

在一个对象/类名之后输入“.”时，代码完成也可以起作用。图A-16展示了一个示例。

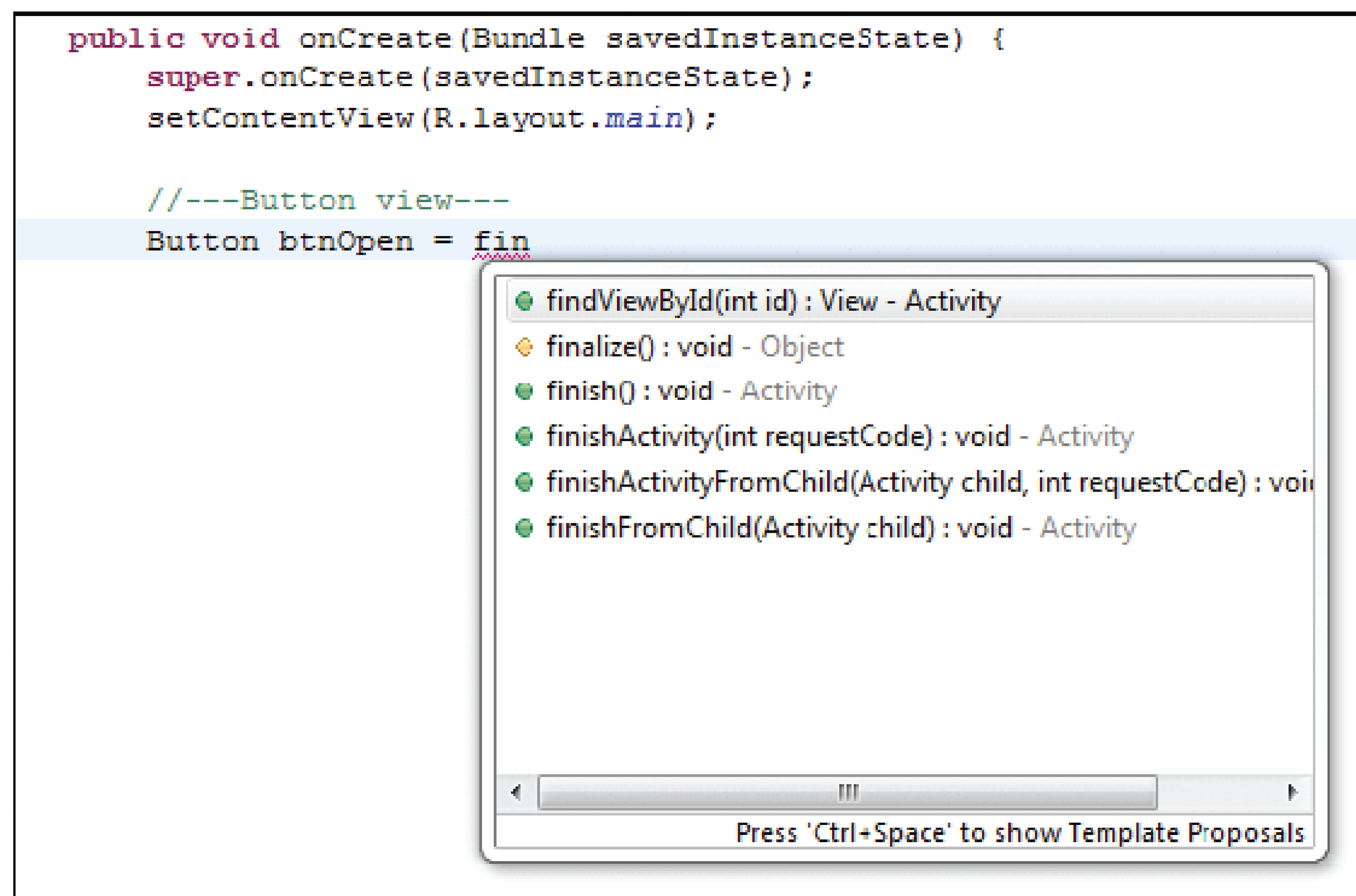


图 A-15

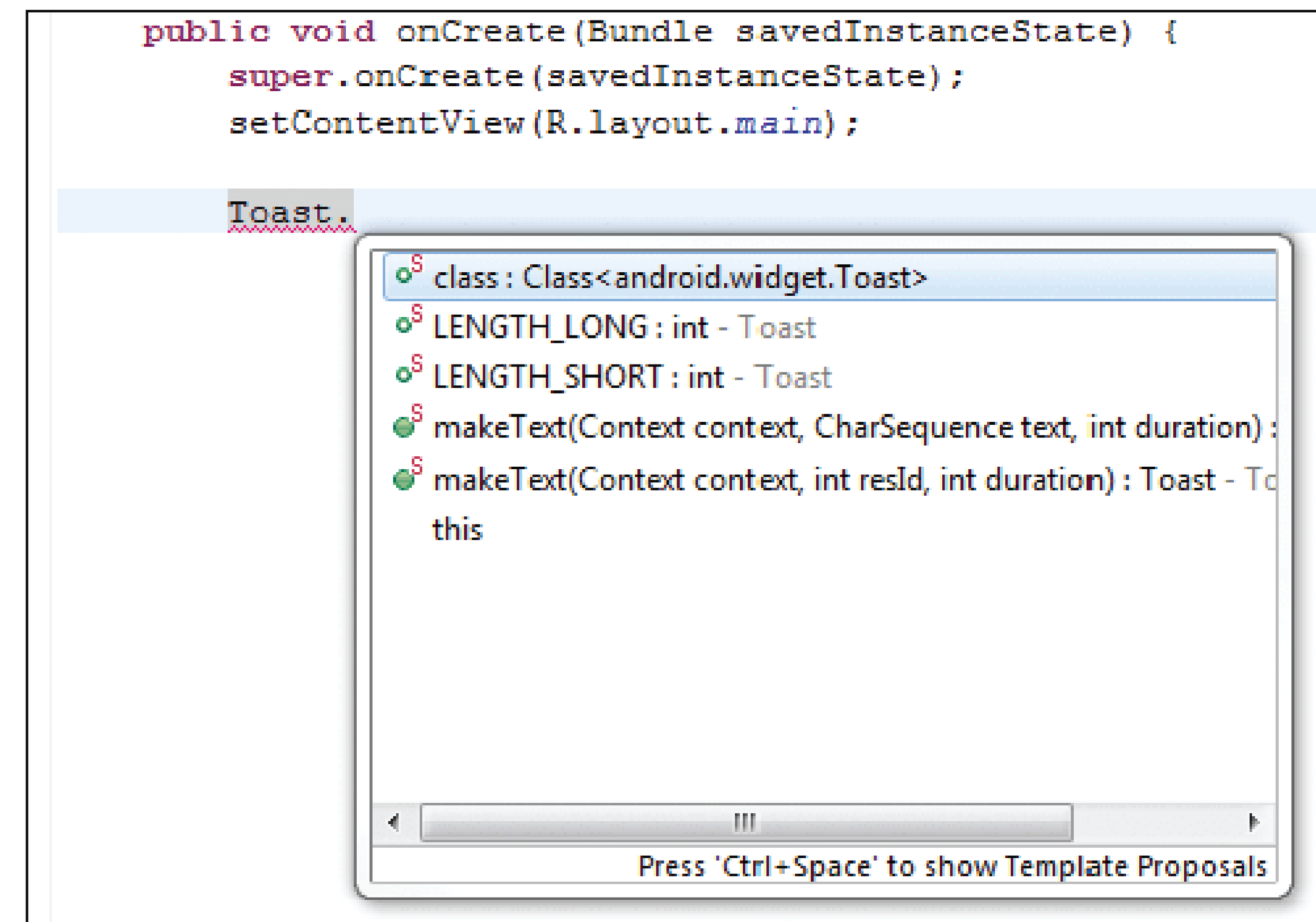


图 A-16

A.1.8 重构

重构是现代IDE支持的一个非常有用的功能。Eclipse支持大量的重构功能，可用来有效地进行应用程序开发。

在Eclipse中，当将光标放到一个特定对象/变量上时，编辑器将突出显示当前源中所选对象出现的所有地方(如图A-17所示)。

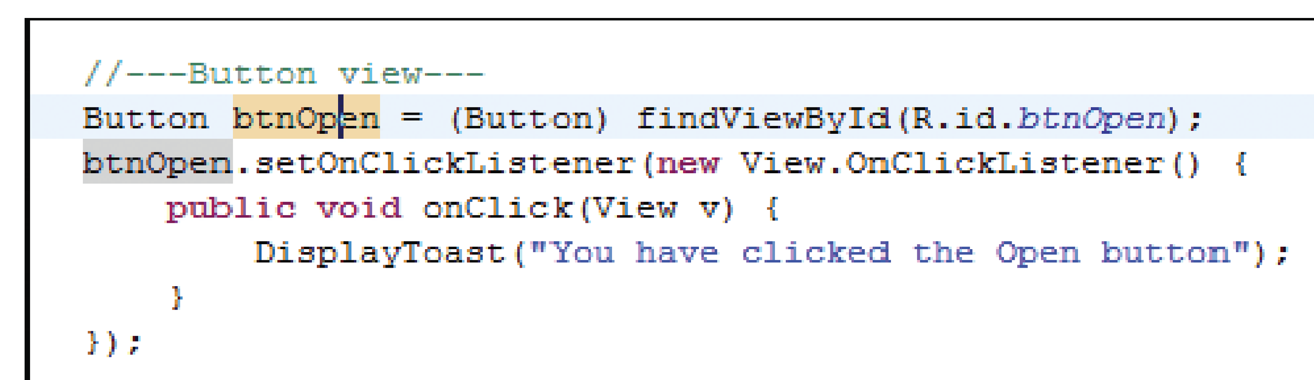


图 A-17

这一功能对于确定一个特定对象在代码中的位置是非常有用的。要改变一个对象的名称，可右击它并选择Refactor | Rename...(如图A-18所示)。

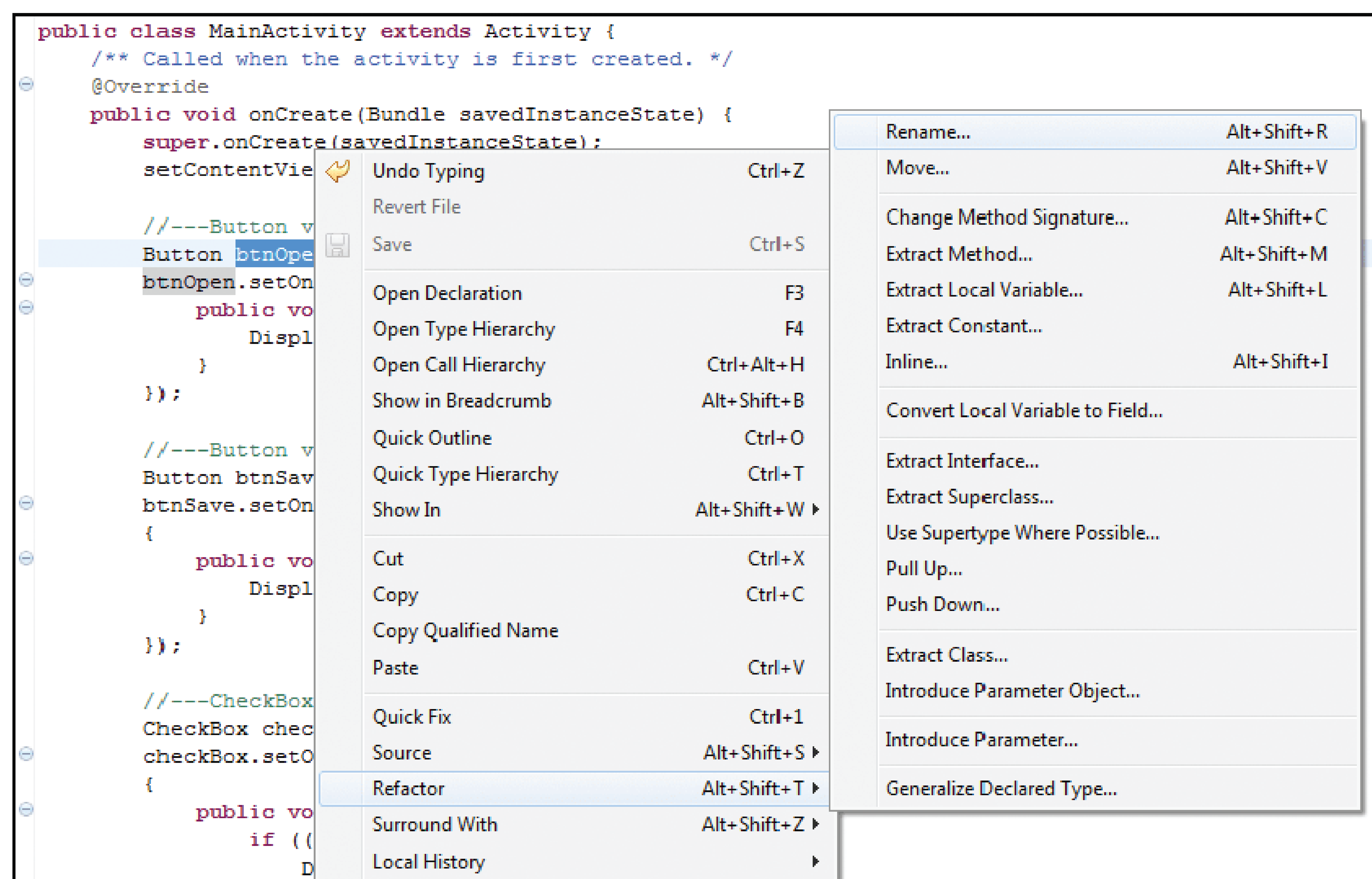


图 A-18

在输入对象的新名称后，该对象出现的所有地方将动态变化(如图A-19所示)。

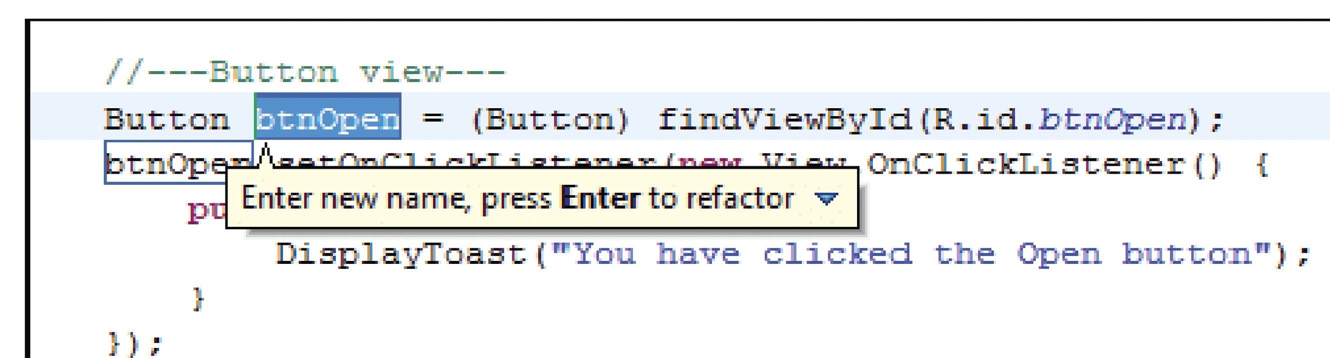


图 A-19

对于重构的详细讨论超出了本书的范围。想要了解Eclipse中更多关于重构的信息，可以访问www.ibm.com/developerworks/library/os-ecref/。

A.2 调试

Eclipse支持在Android模拟器以及真正的Android设备上进行应用程序的调试。当在Eclipse中按下F11键时，Eclipse将首先确定是否已经在运行一个Android模拟器的实例或是连接了一个真实设备。只要有一个模拟器(或设备)运行，Eclipse就会将应用程序部署到运行中的模拟器或已连接的设备上。如果既没有模拟器运行又没有连接设备，Eclipse将自动启动一个Android模拟器的实例并将应用程序部署在其上面。

如果有多个模拟器或设备连接，Eclipse将提示您选择一个目标模拟器/设备，以便在其上部署应用程序(如图A-20所示)。选择一个您想使用的目标设备，然后单击OK按钮。

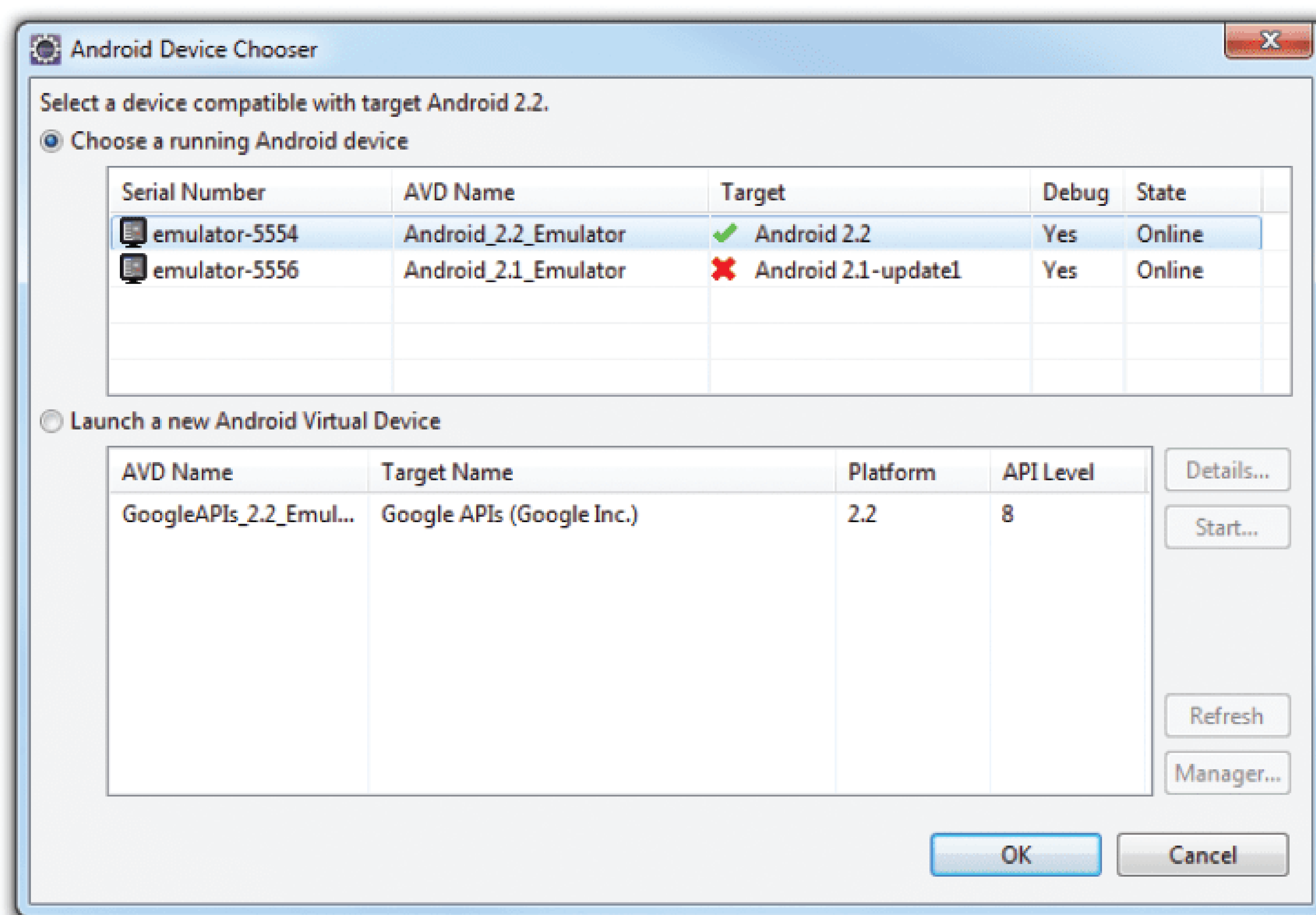


图 A-20

如果想启动一个新的模拟器实例来测试应用程序，可选择Window | Android SDK and AVD Manager来启动AVD管理器。

A.2.1 设置断点

设置断点是临时暂停应用程序的执行，然后检查变量和对象内容的一个好方法。

要设置一个断点，可双击代码编辑器中的最左列。图A-21展示了设置在一个特定语句上的断点。

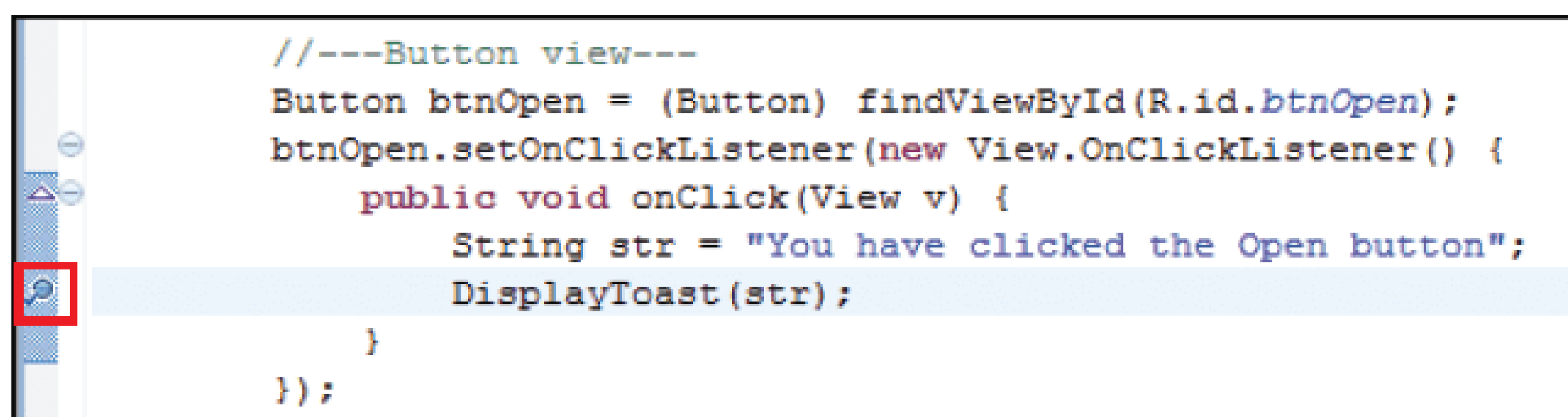


图 A-21

当应用程序运行到第一个断点时，Eclipse将显示一个Confirm Perspective Switch对话框。从根本上说，它希望切换到Debug透视图。为了防止再次出现这一窗口，在底部选中Remember my decision复选框并单击Yes。

现在，Eclipse突出显示了断点(如图A-22所示)。

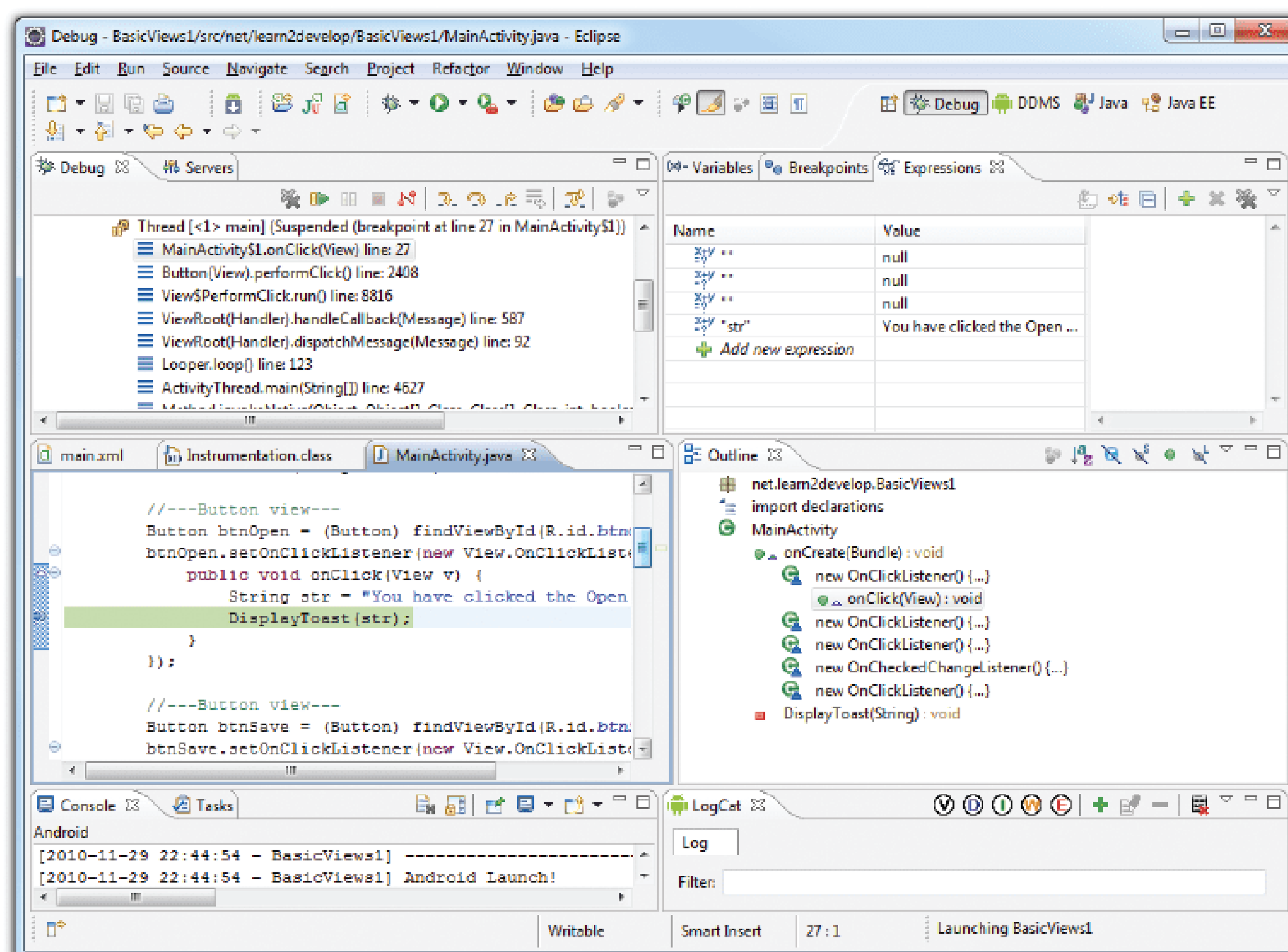


图 A-22

此时，可以利用如图A-23所示的不同的选项(Watch、Inspect和Display)，右击任意选择的对象/变量来查看它们的内容.

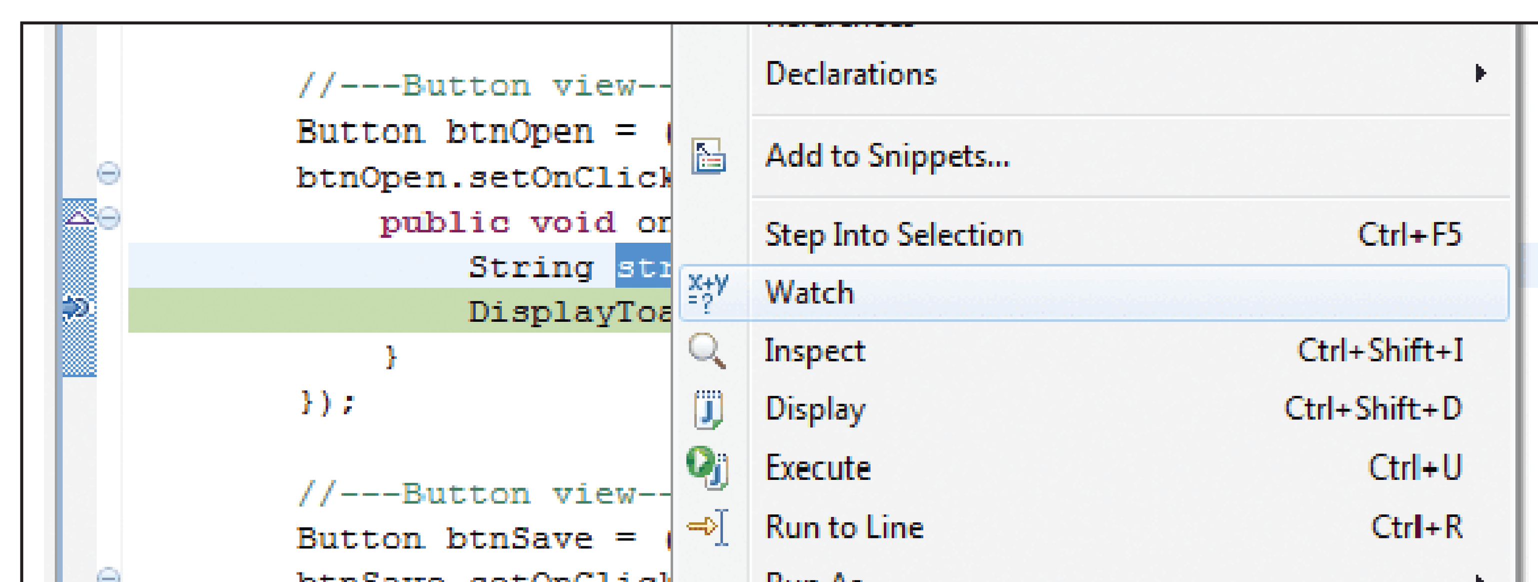


图 A-23

图A-24展示了使用Inspect选项显示str变量的内容。

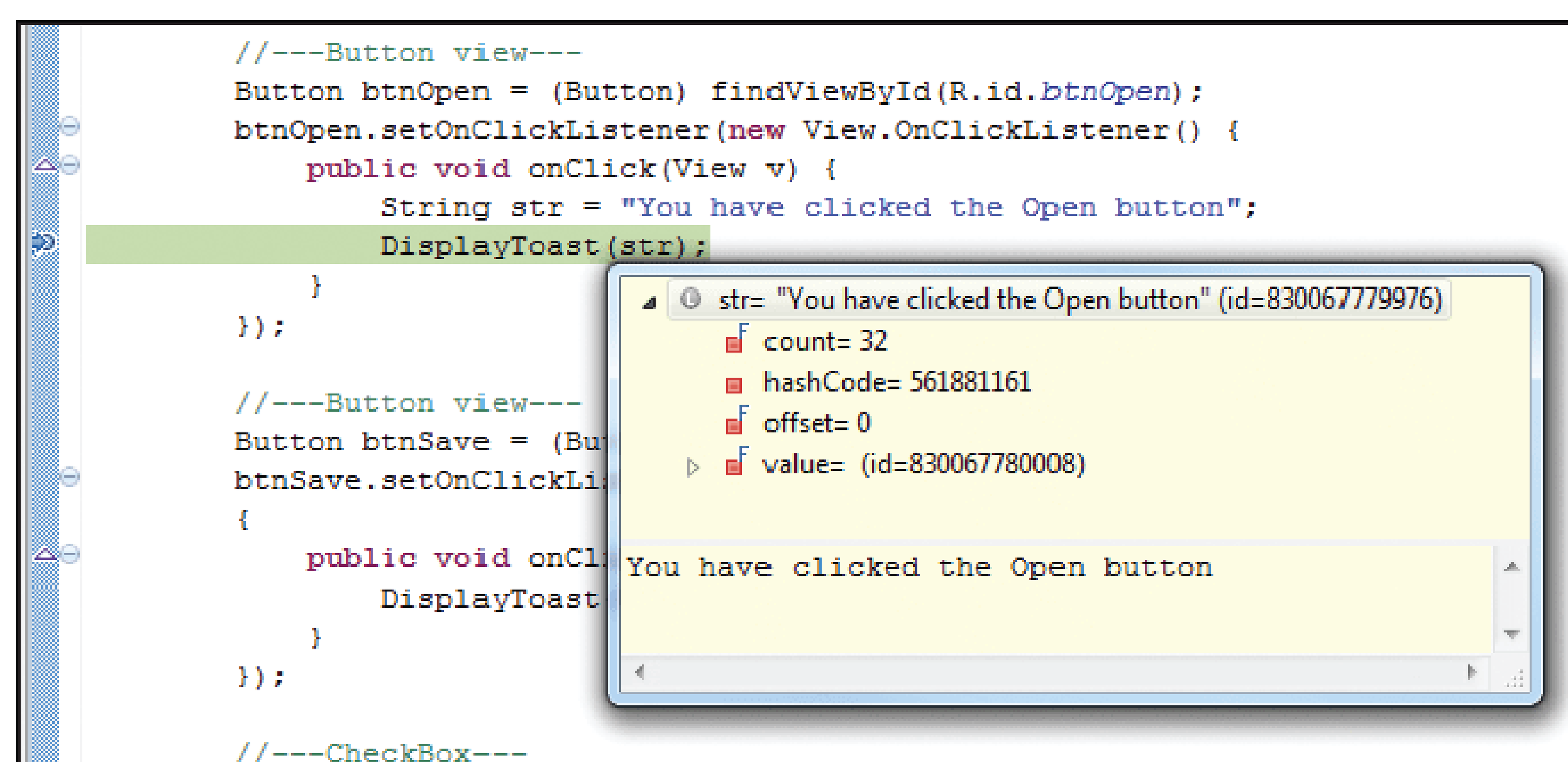


图 A-24

此时，有以下几个选项可以继续执行：

- Step Into——按F5键步进到下一个方法调用/语句。
- Step Over——按F6键跳过下一个方法调用，不进入此方法中。
- Step Return——按F7键从已经进入的方法中返回。
- Resume Execution——按F8键继续执行。

A.2.2 异常

在Android中进行开发时，您将会碰到大量运行时异常，阻止您的程序继续运行。运行时异常的例子包括以下内容：

- 空引用异常(访问一个空对象)
- 没有指定应用程序所需的权限
- 算术运算异常

图A-25展示了一个应用程序产生异常时的当前状态。在这一示例中，我试图从我的应用程序中发送一条SMS消息。当消息刚要发送时，它崩溃了。

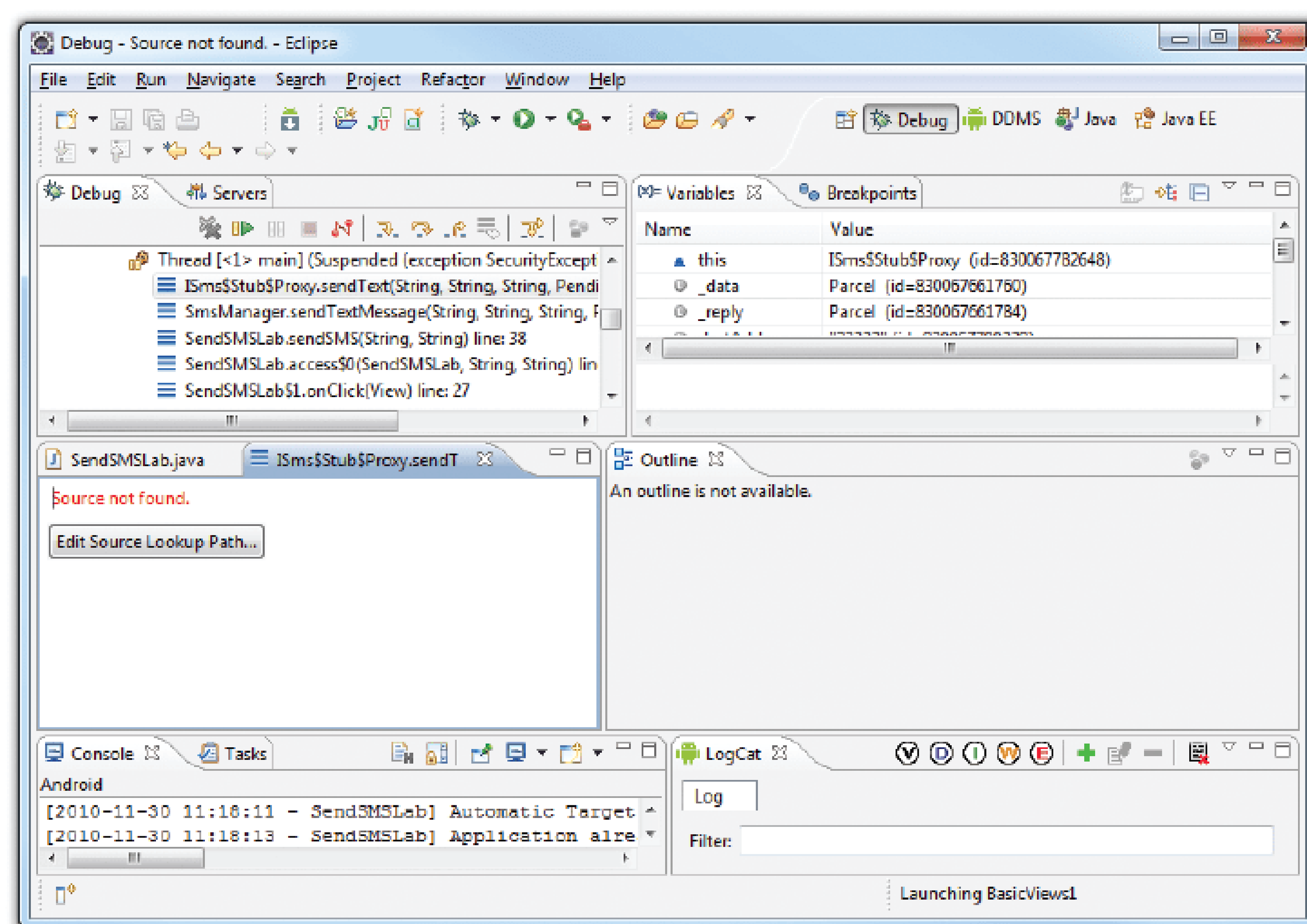


图 A-25

各式的窗口并不能真正识别异常发生的原因。要寻找更多信息，可以在Eclipse中按F6键跳过当前语句。Variables窗口指出了异常的原因，如图A-26所示。在本例中，异常原因是缺少SEND_SMS权限。

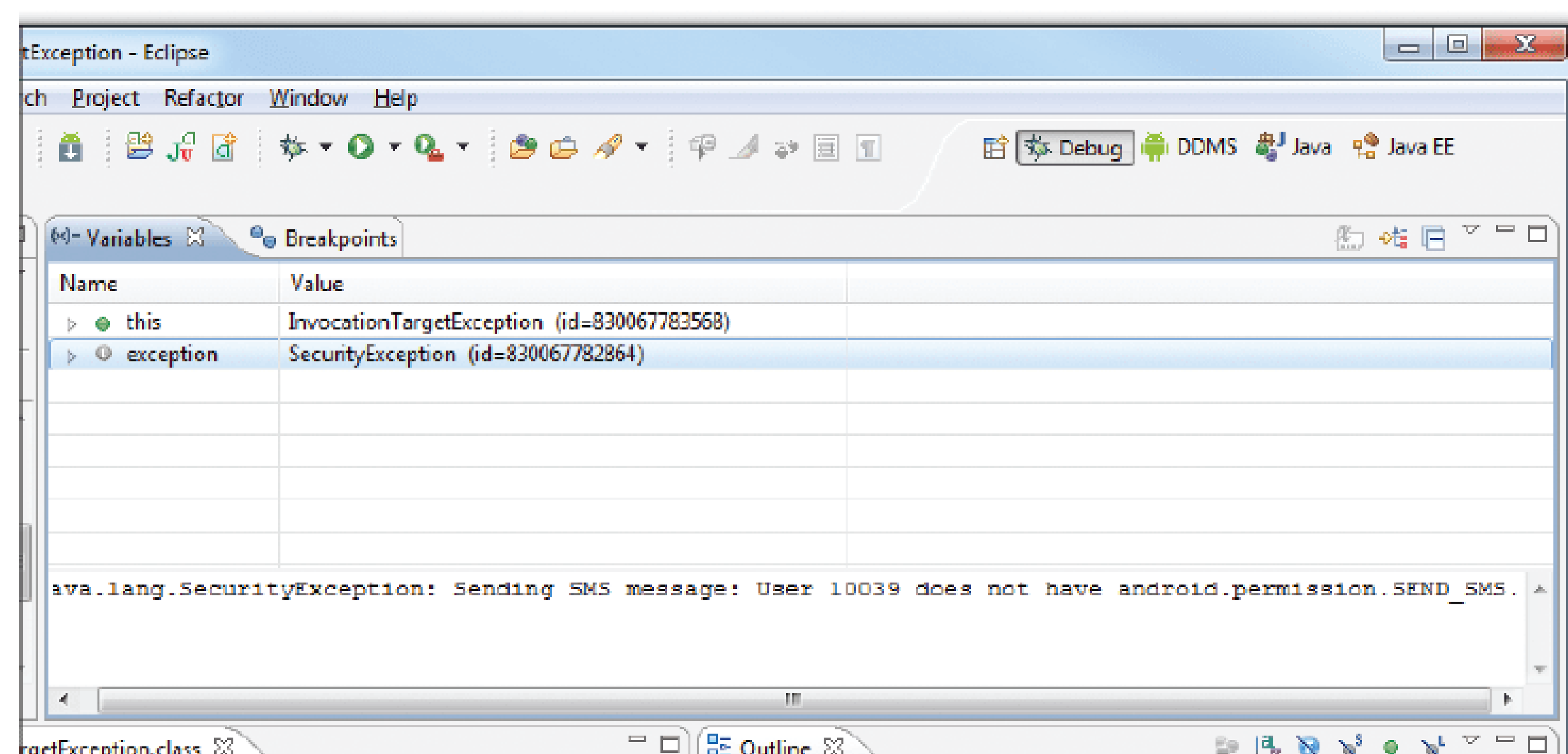


图 A-26

为了补救，只要在AndroidManifest.xml文件中添加以下权限声明就行了：

```
<uses-permission
    android:name="android.permission.SEND_SMS"/>
```

附录 B

使用Android模拟器

Android模拟器自带了Android SDK，它是一个很有用的工具，可以帮助您测试应用程序而不需要购买一台真实设备。虽然您应该在部署应用程序前在真实设备上对其进行彻底的测试，但模拟器还是可以模仿真实设备的大多数功能。模拟器是一个在项目开发阶段应该利用的非常方便的工具。本附录提供了用于掌握Android模拟器的一些常见的提示和技巧。

B.1 Android模拟器的使用

正如在第1章所讨论的，可以使用Android模拟器，通过创建Android Virtual Device(AVD)来模拟不同的Android配置。

启动Android模拟器的方法是在Android SDK and AVD Manager窗口中直接启动创建好的AVD(如图B-1所示)。直接选择AVD并单击Start按钮。可以选择将模拟器缩放到一个特定的大小和显示DPI。

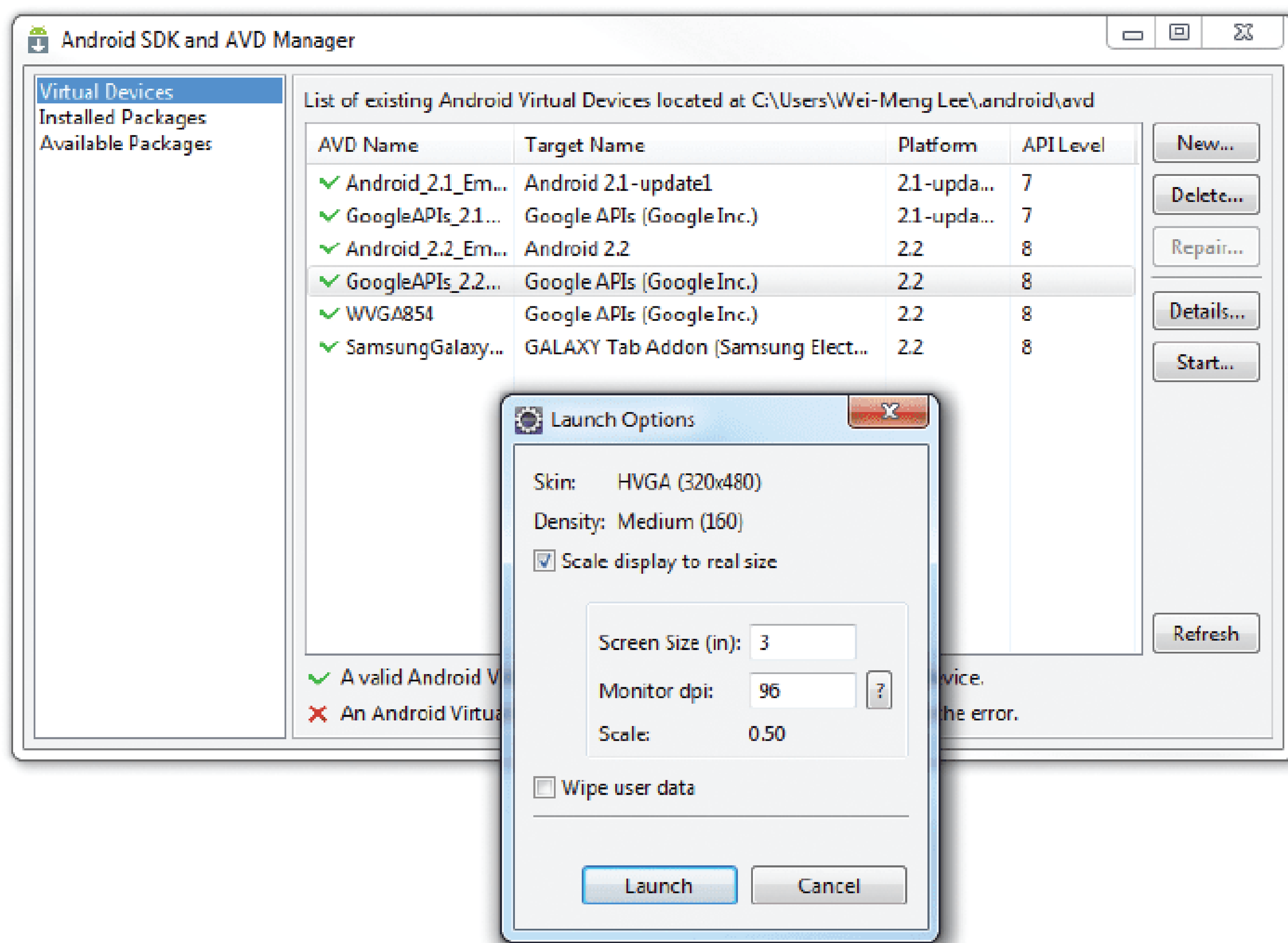


图 B-1

或者，在Eclipse中运行一个Android项目时，自动启动Android模拟器来测试应用程序。可以在Eclipse中为每一个Android项目定制Android模拟器，只要选择Run | Run Configurations。选择

左边位于Android Application下的项目名称(如图B-2所示)，就会在右边看到Target选项卡。在其中可以选择用来进行应用程序测试的AVD，以及选择模拟不同的场景，如网速和网络延迟。

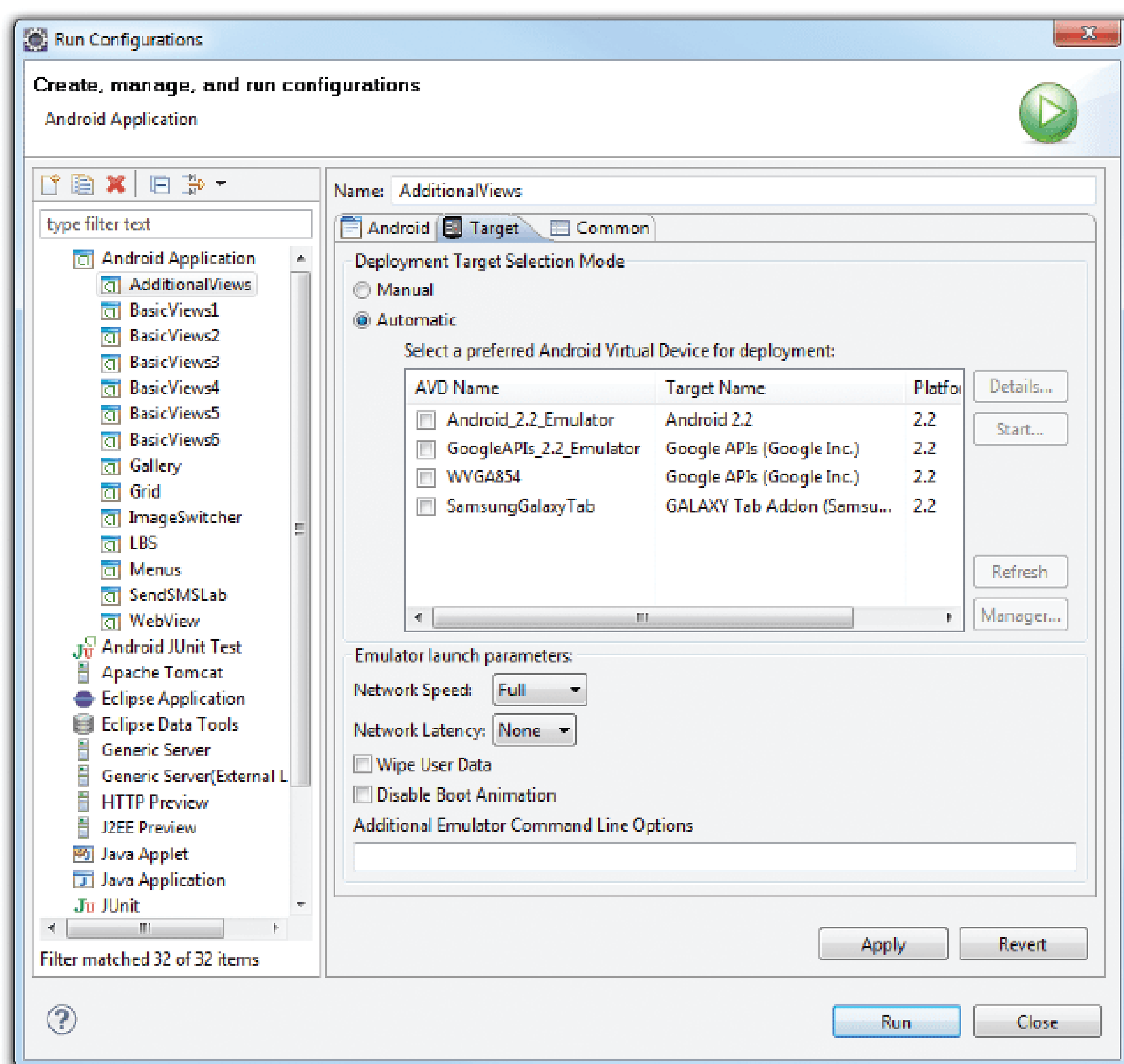


图 B-2

B.2　安装自定义的AVD

有时，设备制造商提供了它们自己的AVD，可用于模拟在它们的设备上运行应用程序。三星公司就是一个好的示例，它提供了Samsung Galaxy Tab(http://innovator.samsungmobile.com/galaxyTab.do)附加组件来模拟它们的Samsung Galaxy Tab平板电脑。要安装Samsung Galaxy Tab附加组件，首先在Eclipse中启动Android SDK and AVD Manager，然后在对话框左侧选择Available Packages选项(如图B-3所示)。

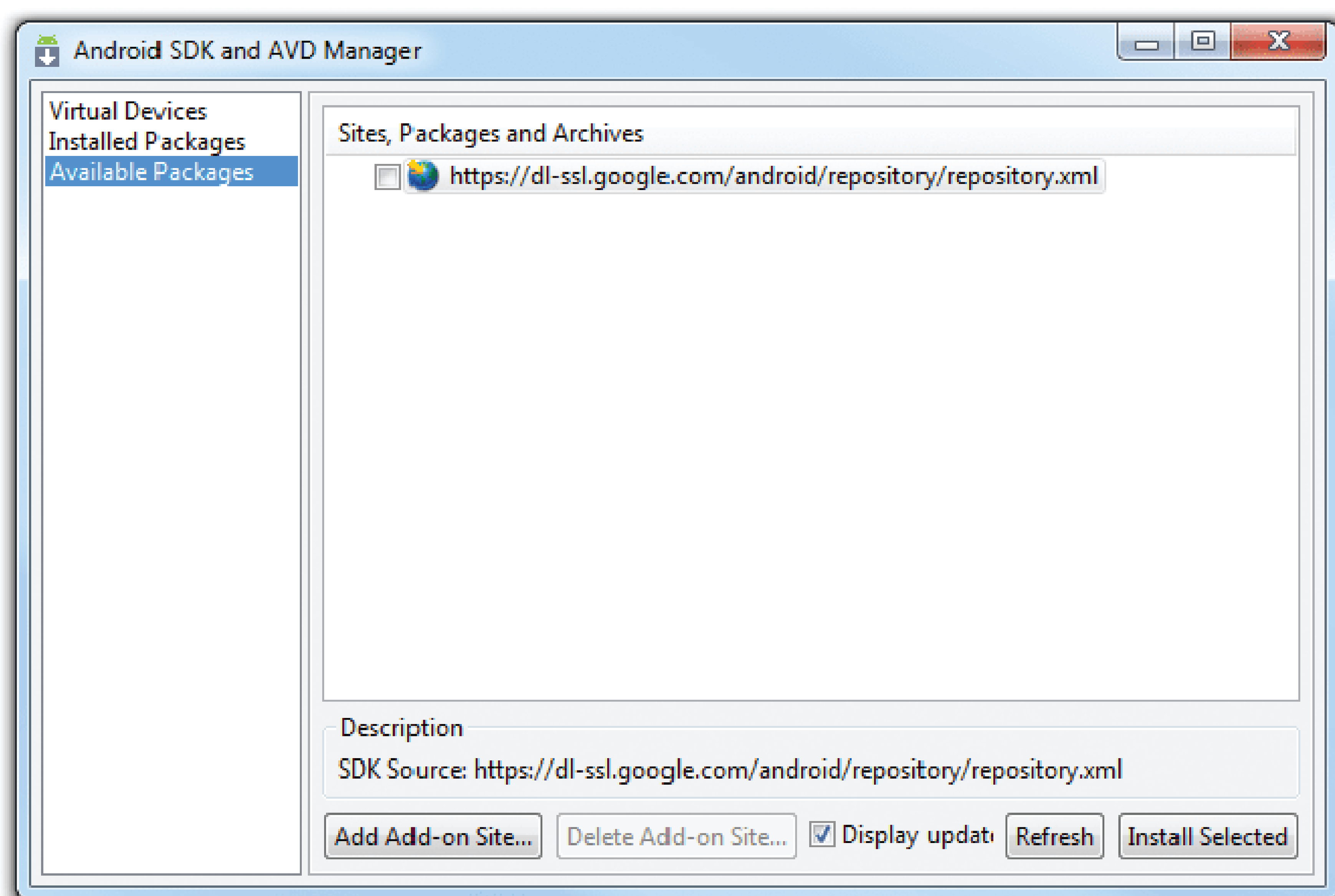


图 B-3

在屏幕底部，单击Add Add-on Site...按钮并输入以下URL: http://innovator.samsungmobile.com/android/repository/srepository.xml(如图B-4所示)，然后单击OK按钮。

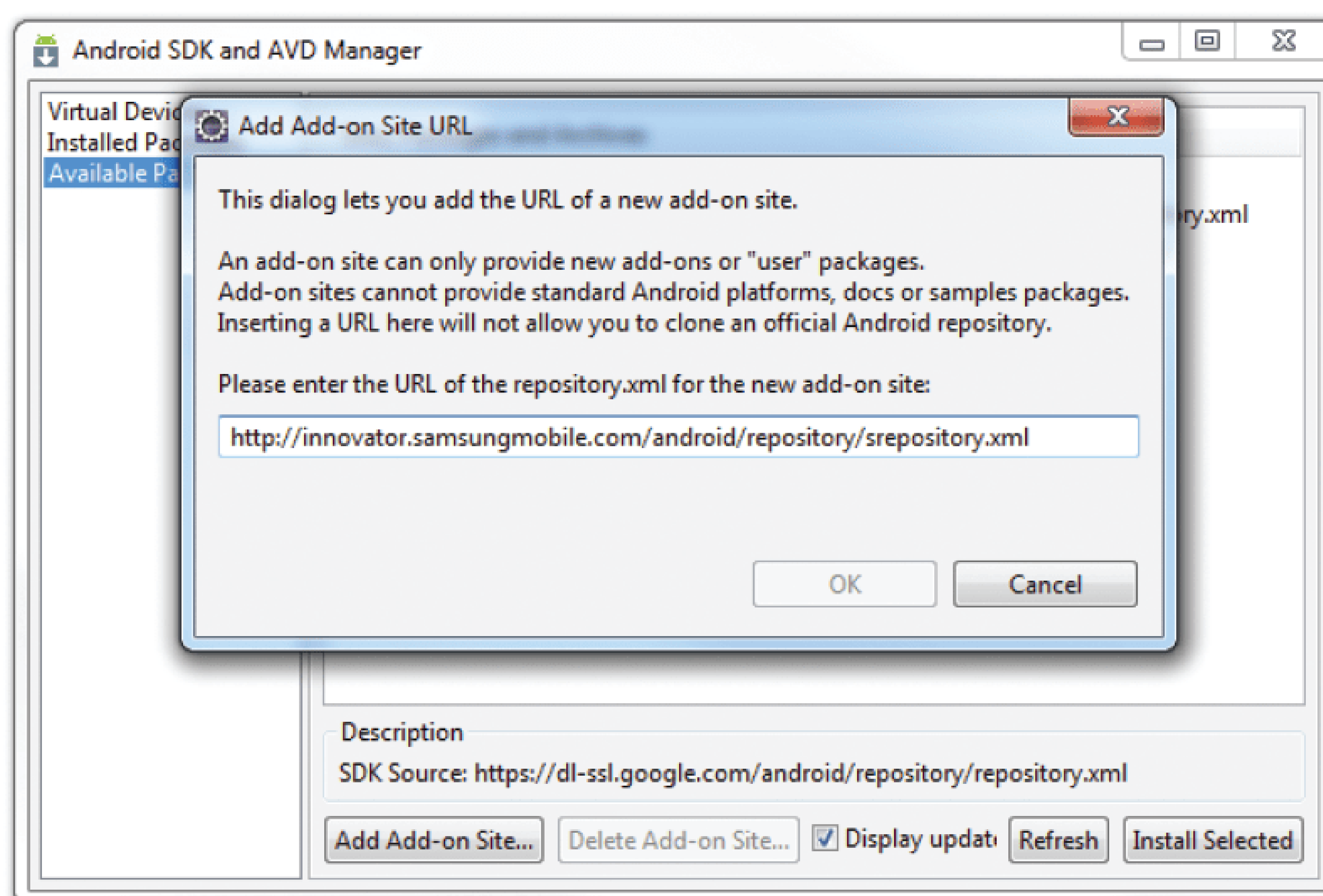


图 B-4

现在您应该看到有额外的包可用了(如图B-5所示)。选中此包并单击Install Selected按钮。

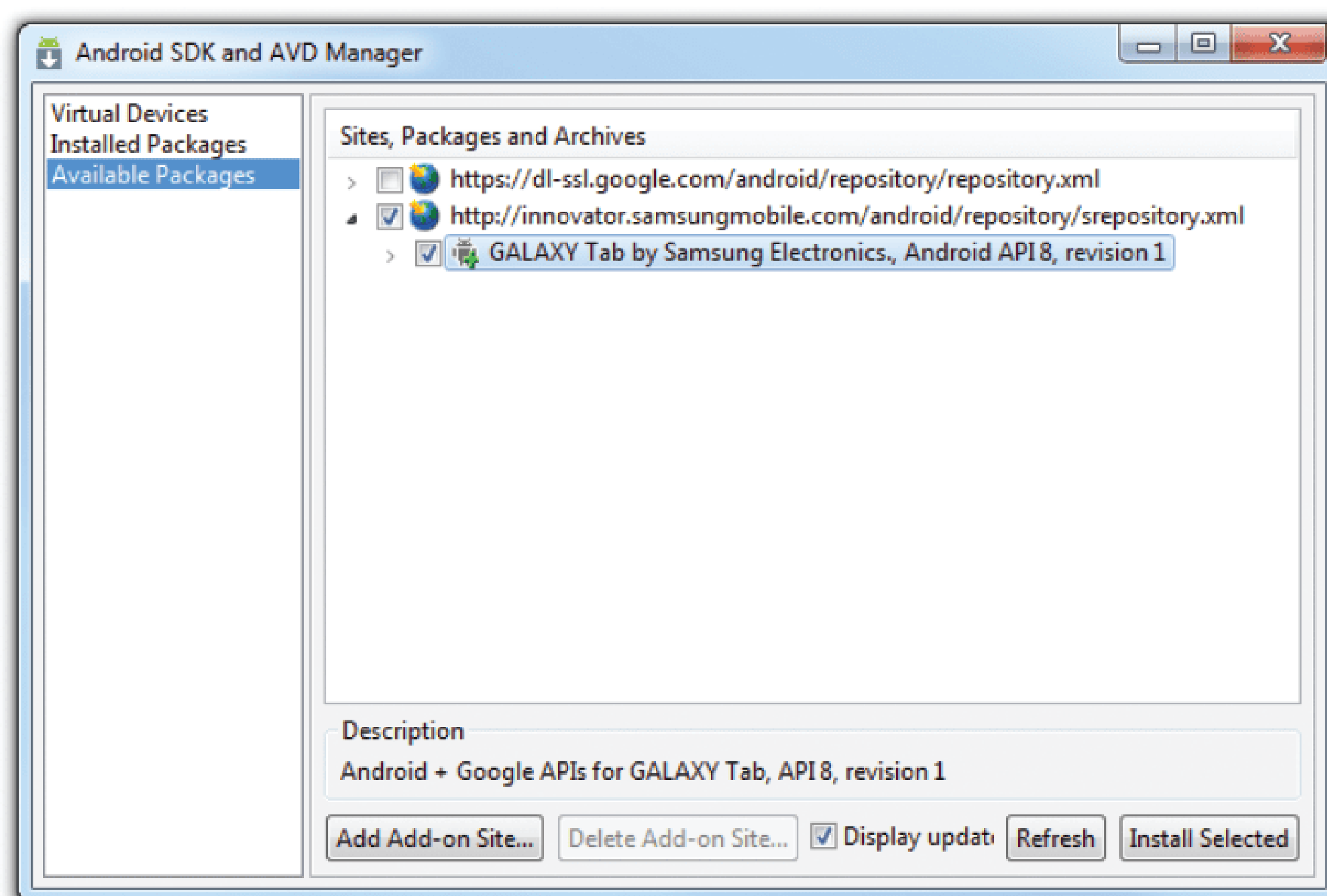


图 B-5

在弹出的对话框中，单击Accept接受许可协议，然后单击Install来下载并安装包。

当下载的包安装完毕，您就可以基于最新下载的包创建一个新的AVD。在Android SDK and AVD Manager窗口中选择Virtual Devices项并单击New按钮。

按图B-6所示为新的AVD命名。单击Create AVD按钮来创建这个AVD。

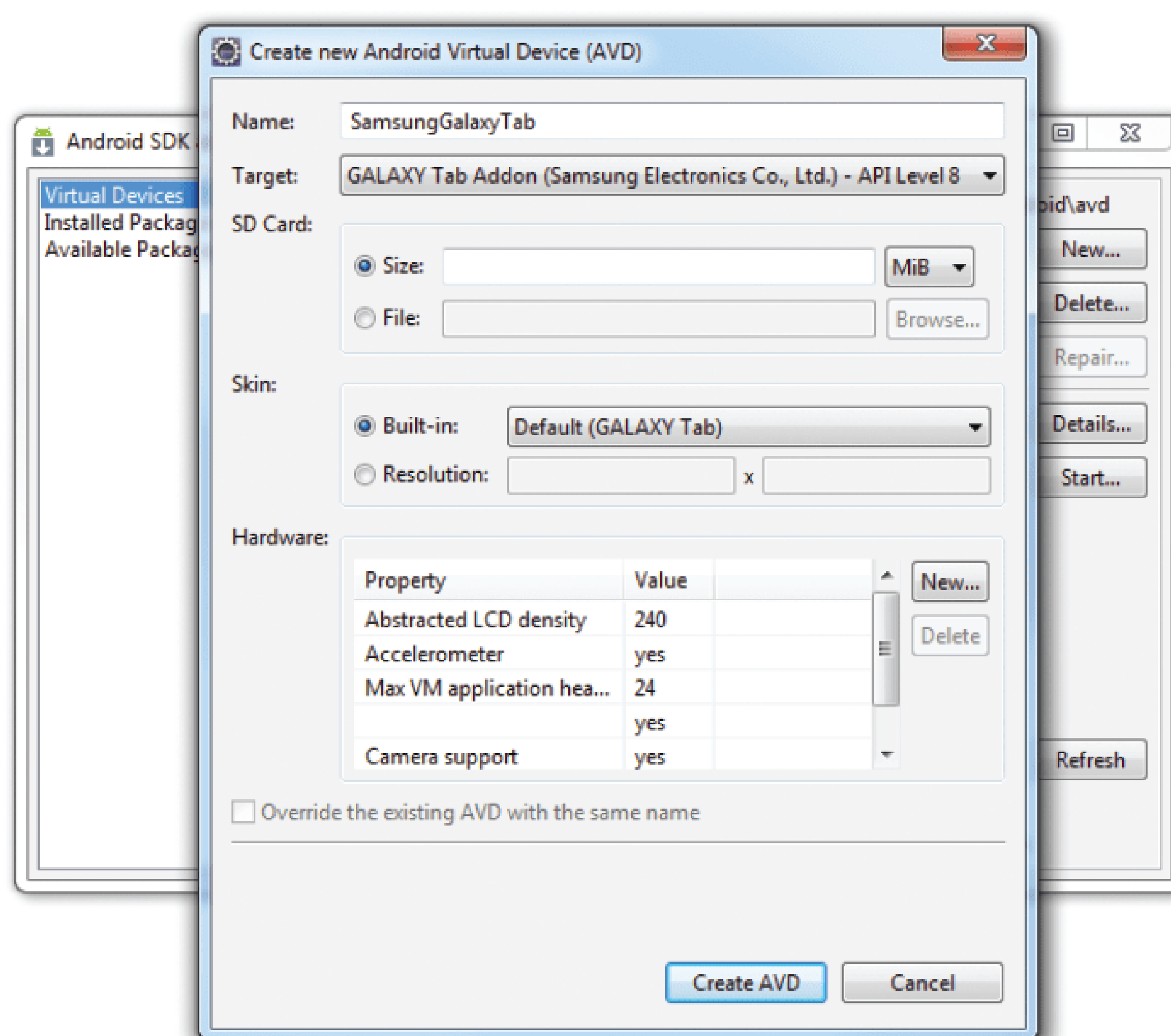


图 B-6

要启动SamsungGalaxyTab AVD，可选中它并单击Start...按钮，这时将显示Launch Options对话框(如图B-7所示)。

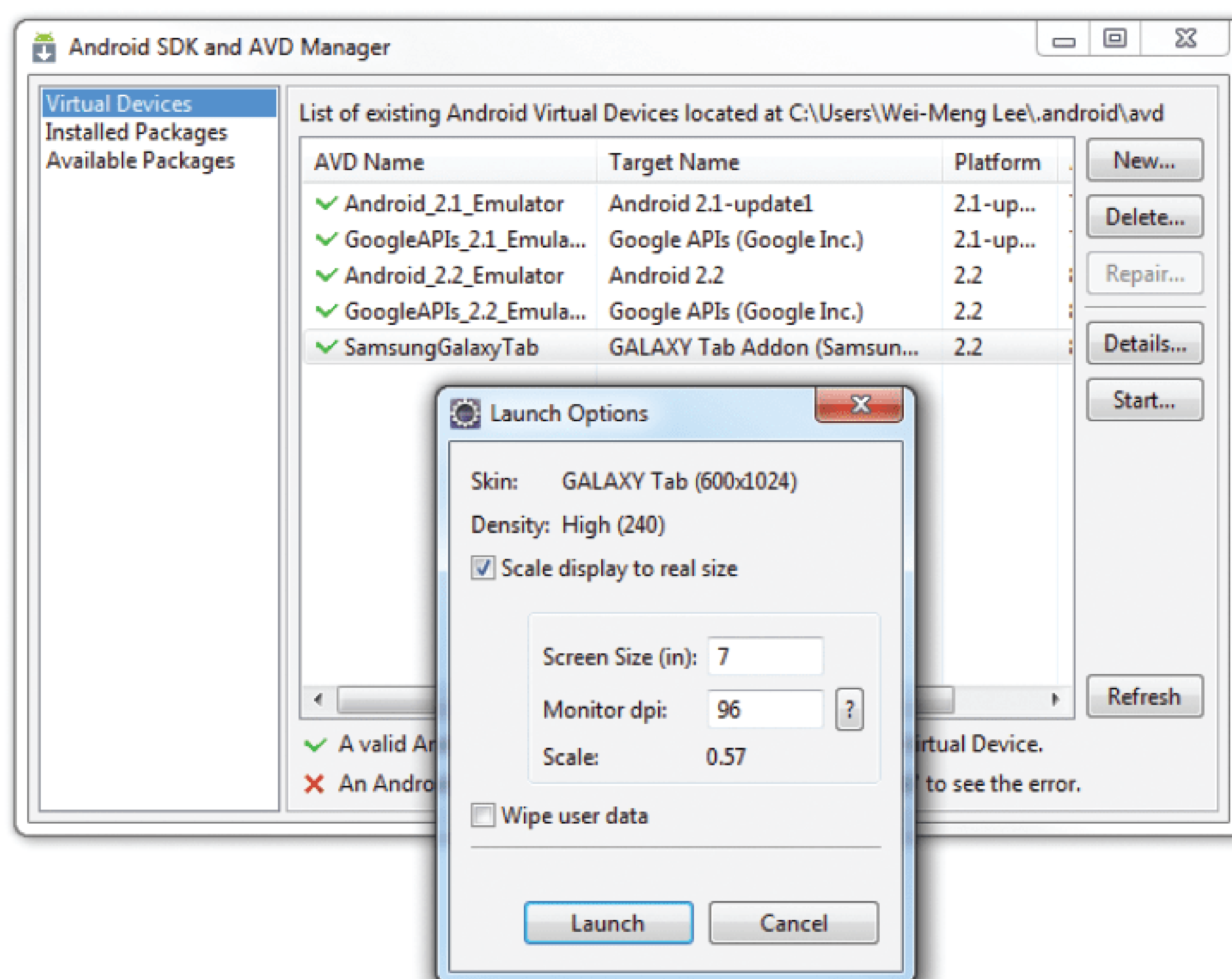


图 B-7

如果想要重新调整模拟器的大小，可选中Scale display to real size选项。如果您在一个小的显示器上运行模拟器的话(例如在一个笔记本电脑上)，这就非常有用了。指定屏幕大小并单击Launch按钮来启动模拟器。图B-8展示了Samsung Galaxy Tab模拟器。

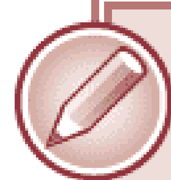


图 B-8

B.3　模拟真实设备

除了使用Android模拟器来测试Android的不同配置以外，还可以使用由设备制造商提供的系统映像，利用模拟器来模拟真实设备。

例如，HTC为其运行Android1.5和1.6的设备提供了映像(http://developer.htc.com/google-io-device.html#s3)。您可以下载一个设备的系统映像，然后使用该系统映像利用Android模拟器来模拟真实设备。这里将讲述如何做到这一点(理论上，这对任何版本的Android都有效)。



注意： 如果使用HTC的映像，您应该能够启动模拟器，这是毫无问题的。然而，它不能启用网络。一些好心人上传了一个修改后的运行得很好的映像。您可以试着在www.4shared.com/get/x6pZm3-W/system.html上下载。

首先，使用Android SDK and AVD Manager，创建一个新的AVD。在HTC这个示例中，利用Android 1.6作为平台创建一个AVD。这个AVD位于C:\Users\<*username*>\.android\avd\<*avd_name*>.avd文件夹下。如图B-9所示，新创建的AVD在该文件夹中只包含两个文件。

使用下载的系统映像，将system.img文件复制到AVD文件夹下，如图B-10所示。

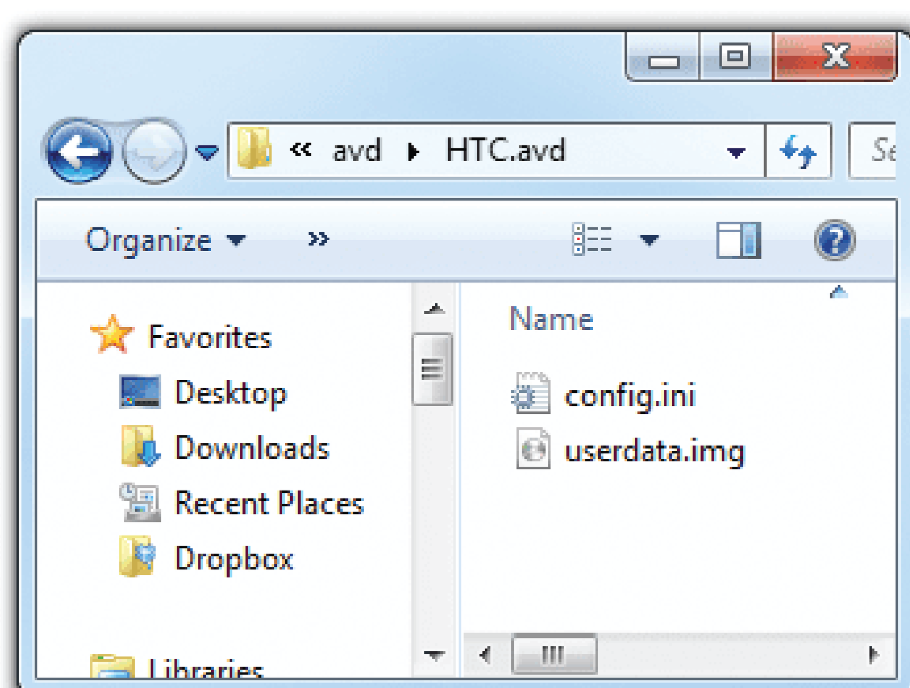


图 B-9

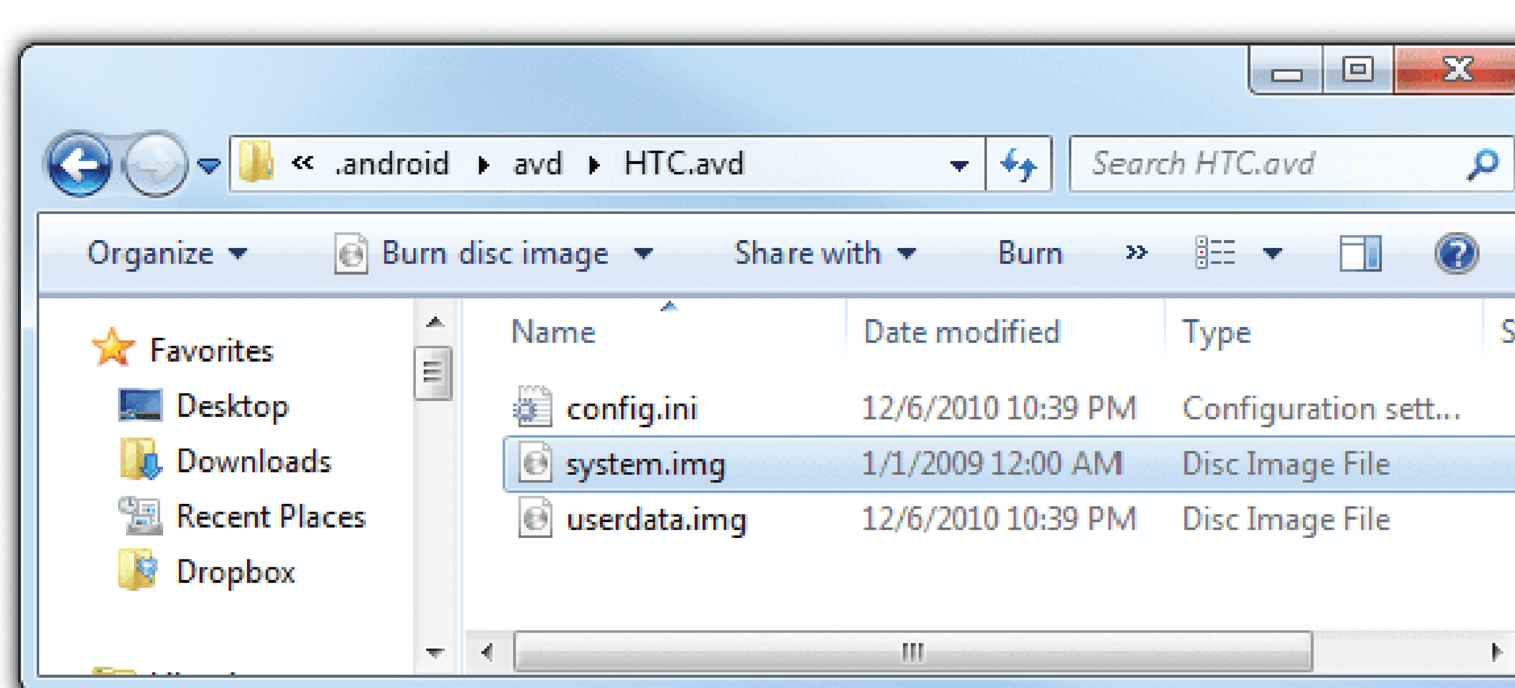


图 B-10

启动AVD，应该可以看到它正在启动中(如图B-11所示)。

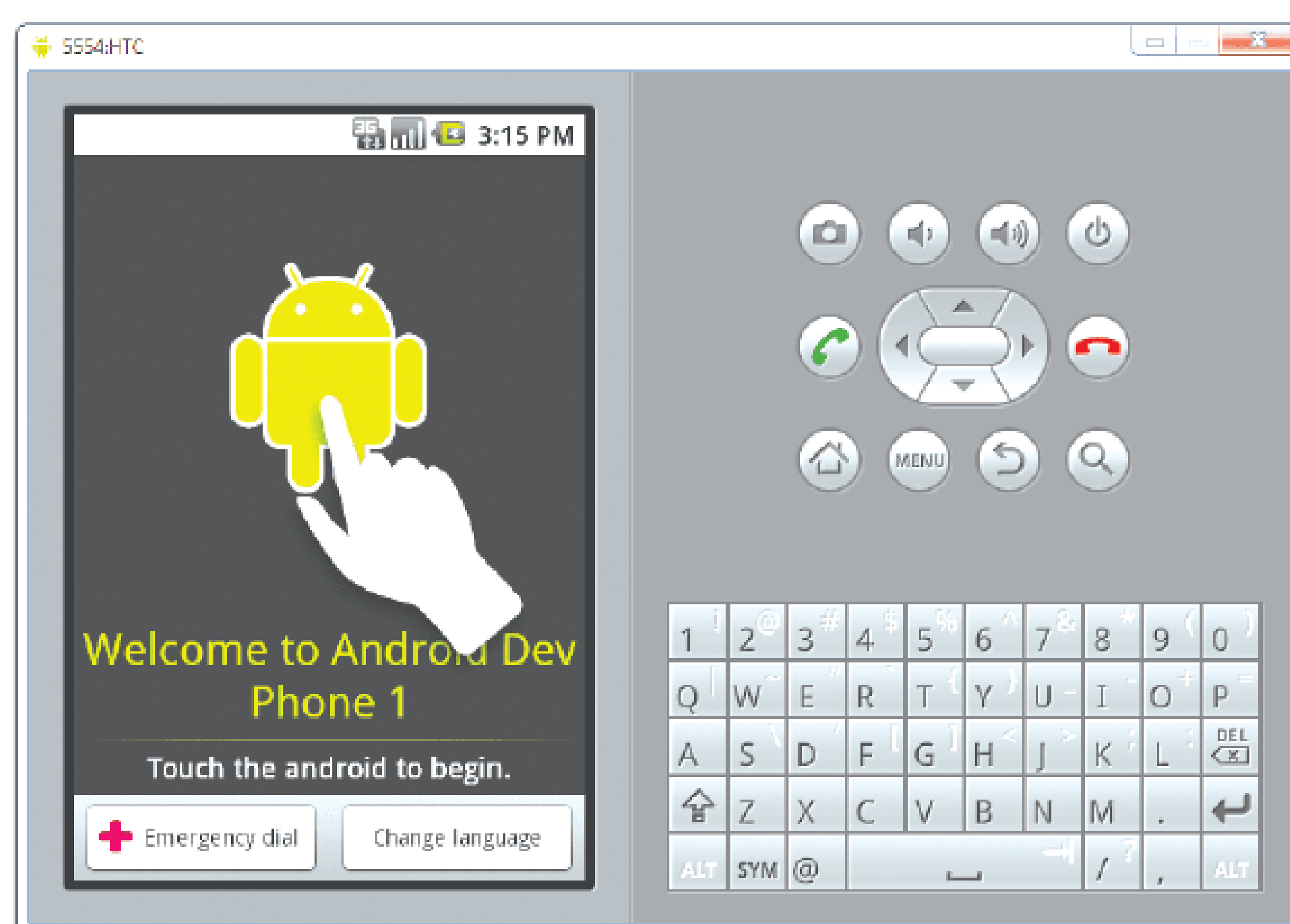


图 B-11

您可以使用您的Google账户进行登录(如图B-12所示)。当提示滑动屏幕以打开键盘时，按下Ctrl+F11组合键改变模拟器的方向。这个动作可使模拟器相信您正在滑动屏幕来打开键盘。

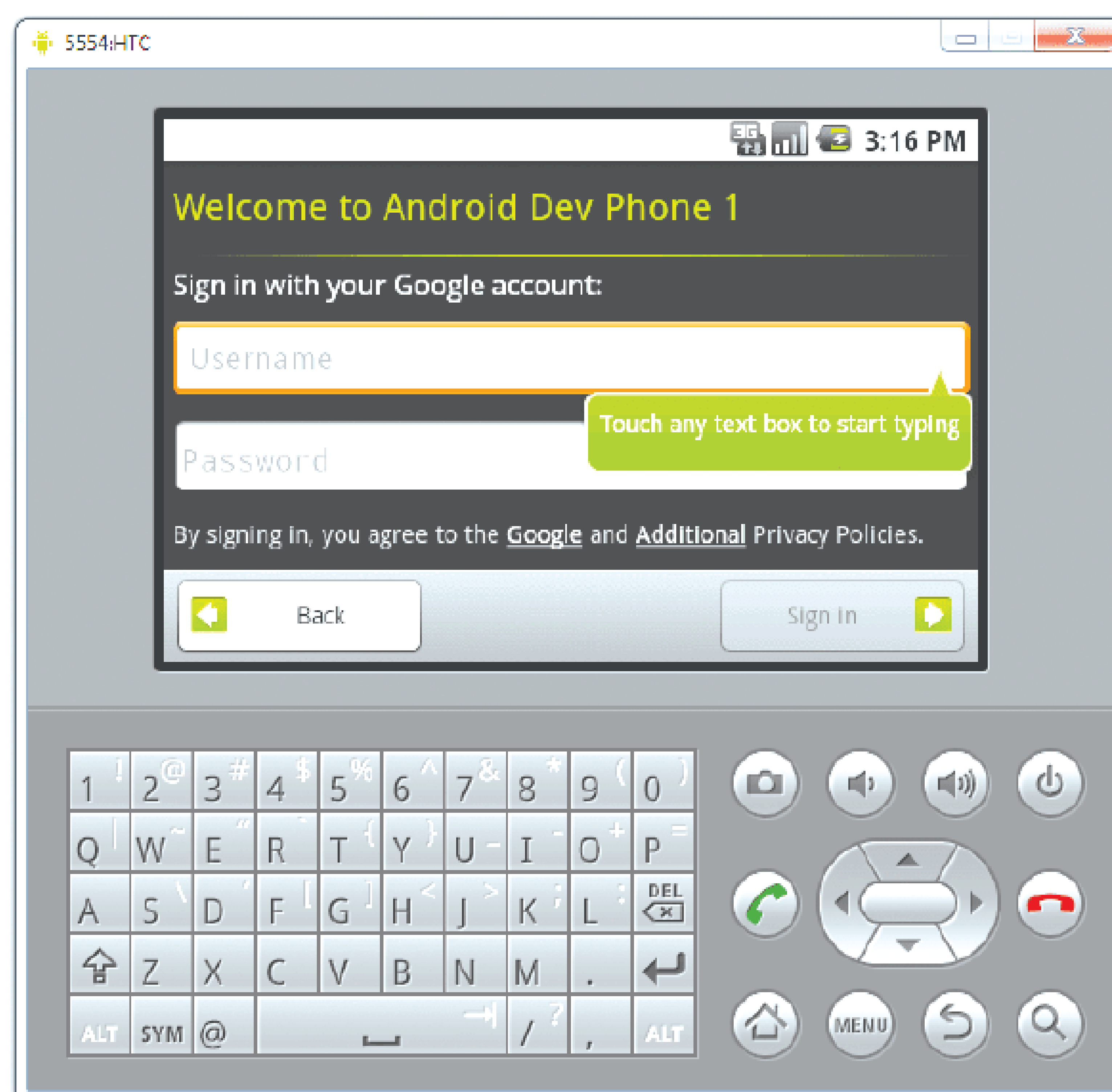


图 B-12

一旦登录成功，您将能够在模拟器上浏览Android Market(如图B-13所示)。

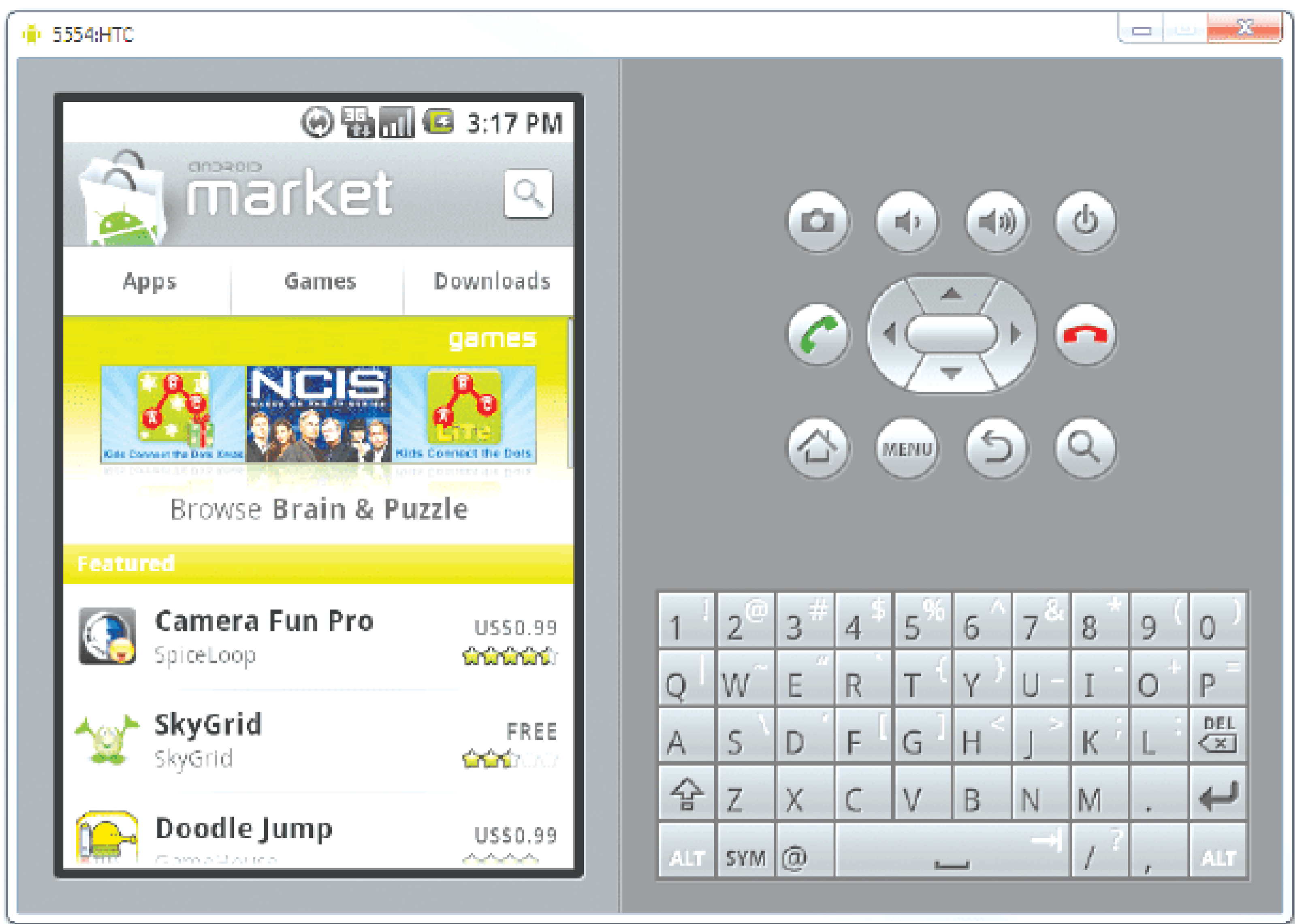


图 B-13

B.4　模拟SD卡

如果创建了一个新的AVD，那么可以模拟存在着一张SD卡(如图B-14所示)。直接输入想模拟的SD卡的大小(图中，大小为200MiB)。

或者，可以通过首先创建一个磁盘映像并将其附加到AVD上，来模拟Android模拟器中的一张SD卡。mksdcard.exe实用程序(也位于Android SDK的tools文件夹下)可以用来创建一个ISO磁盘映像。下列命令创建一个大小为2GB的ISO映像(还可参见图B-15):

```
mksdcard 2048M sdcard.iso
```

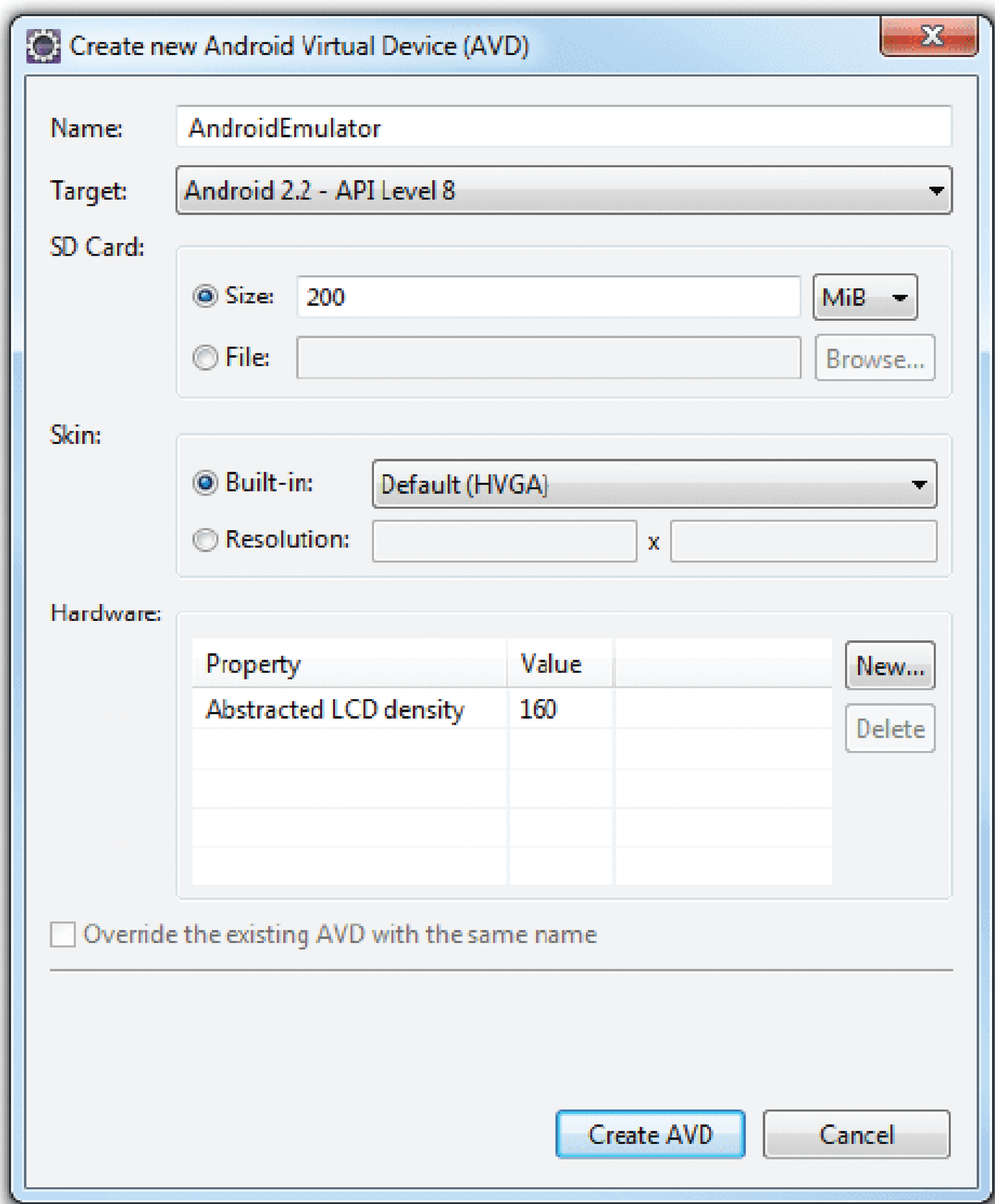


图 B-14

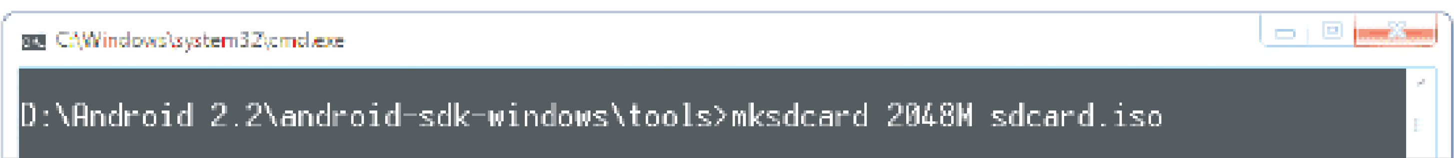


图 B-15

一旦创建了映像，就可以指定ISO文件的位置了，如图B-16所示。

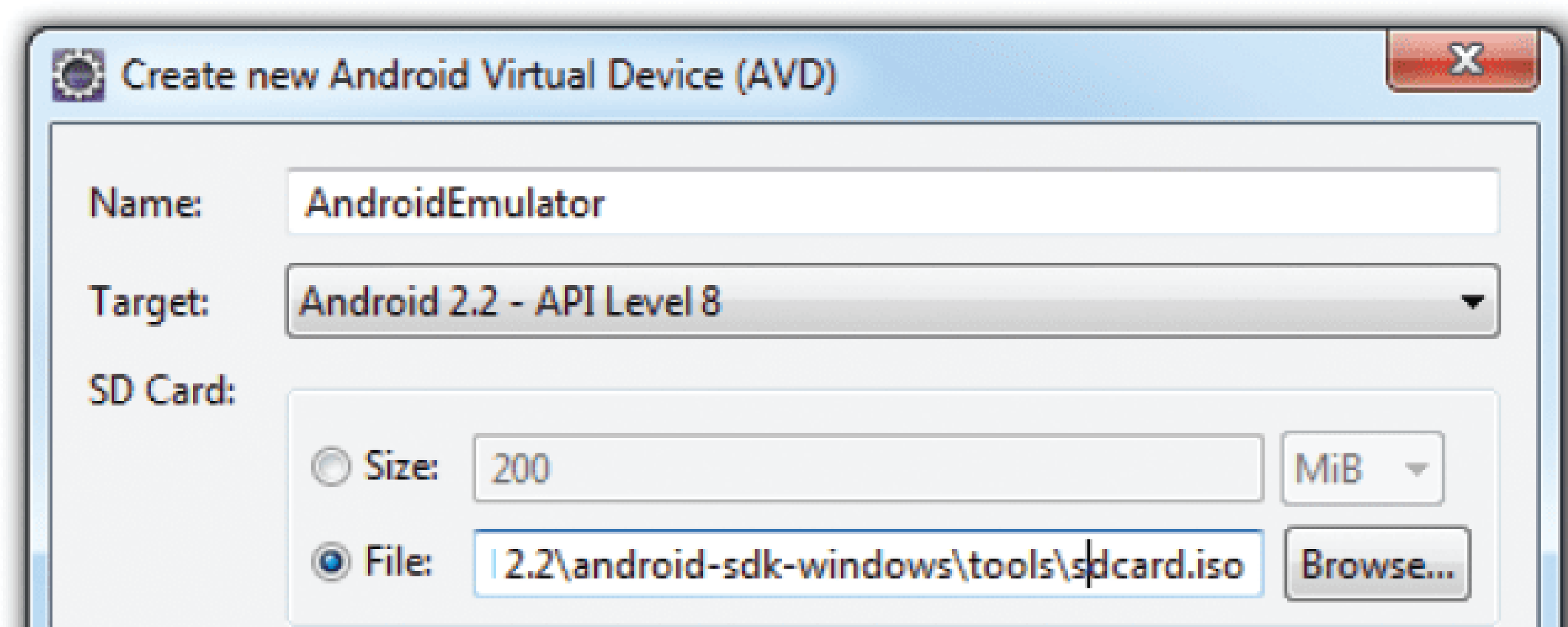


图 B-16

B.5　模拟具有不同屏幕大小的设备

除了模拟SD卡外，还可以模拟具有不同屏幕大小的设备。图B-17展示了AVD在模拟WVGA854外观，其分辨率是480×854 像素。注意，LCD的像素密度是240，意思是该屏幕每英寸具有240个像素。

对于您选择的每一个目标，都有一个可用的外观列表。Android SDK支持以下屏幕分辨率：

- QVGA —— 240×320
- WQVGA400 —— 240×400
- WQVGA432 —— 240×432
- HVGA ——320×480
- WVGA800 —— 480×800
- WVGA854 —— 480×854

图B-18展示了使用WVGA854外观的Android模拟器。

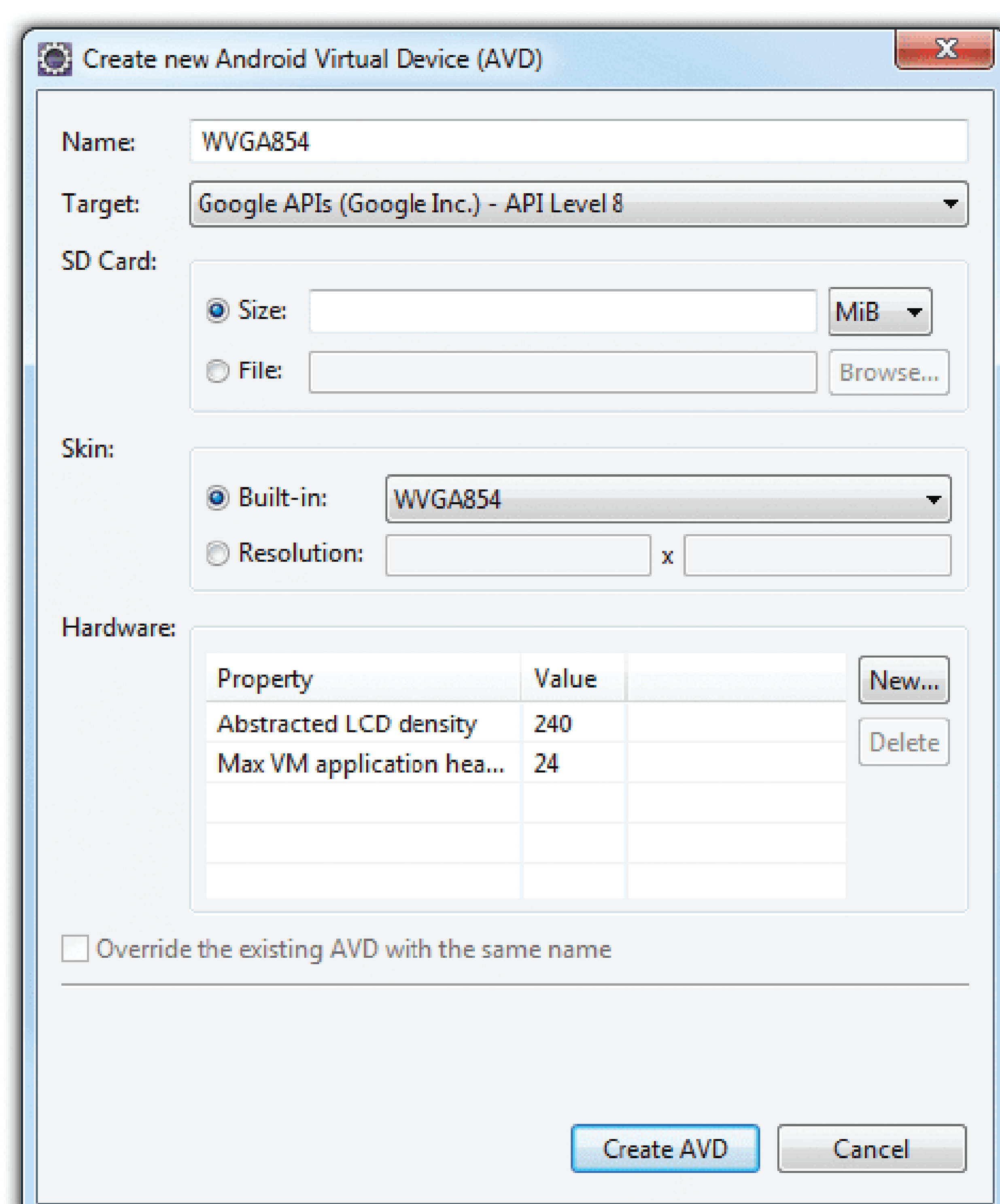


图 B-17

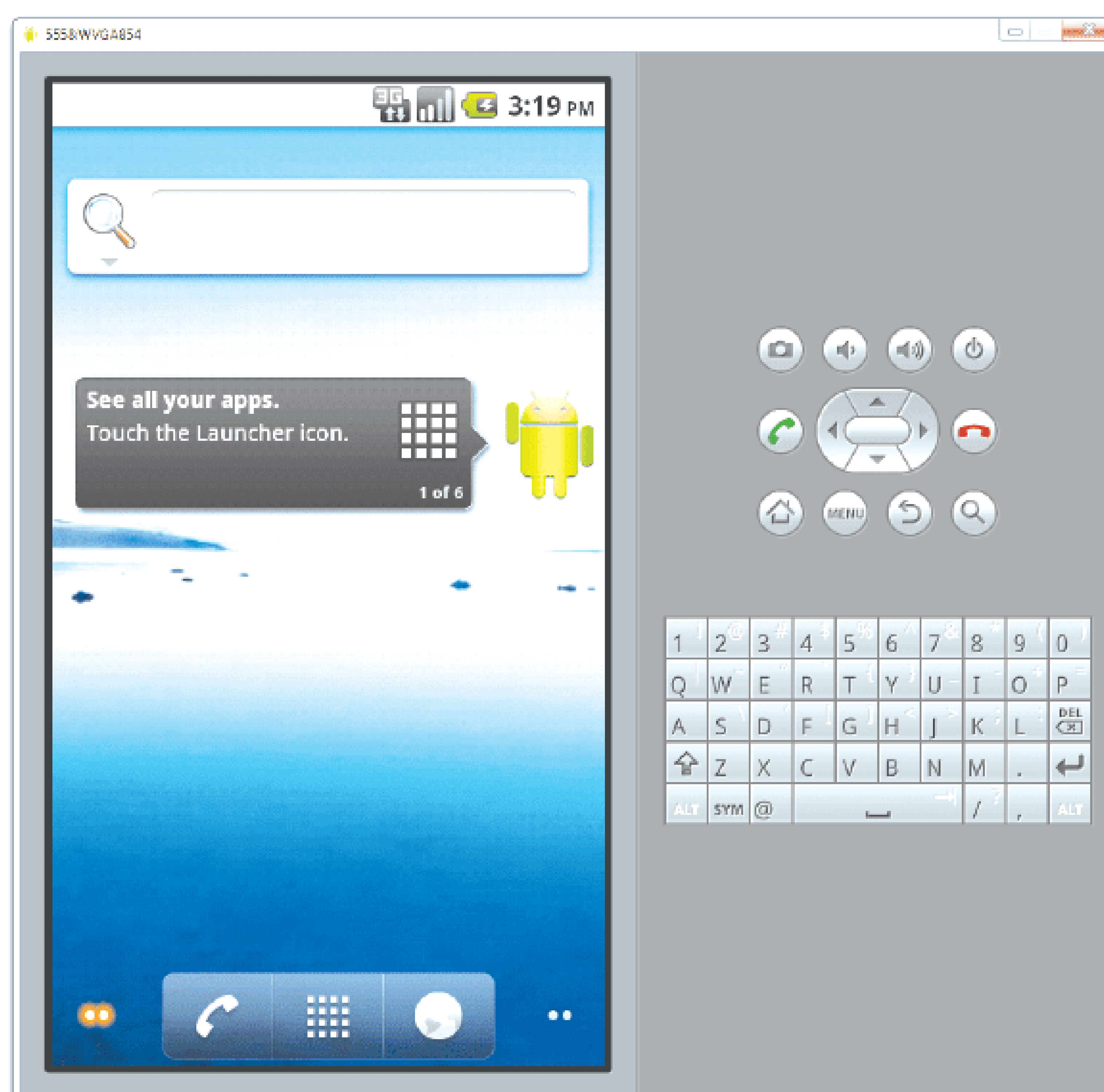


图 B-18

B.6　模拟物理功能

除了模拟具有不同屏幕大小的设备之外，还可以选择模拟不同的硬件功能。当创建一个新AVD时，单击New按钮将显示一个可用于选择打算模拟的硬件的类型的对话框(如图B-19所示)。

例如，如果想模拟一个没有触摸屏的Android设备，可以选择Touch-screen support属性并单击OK按钮。回到AVD对话框，将该属性值从yes变为no(如图B-20所示)。

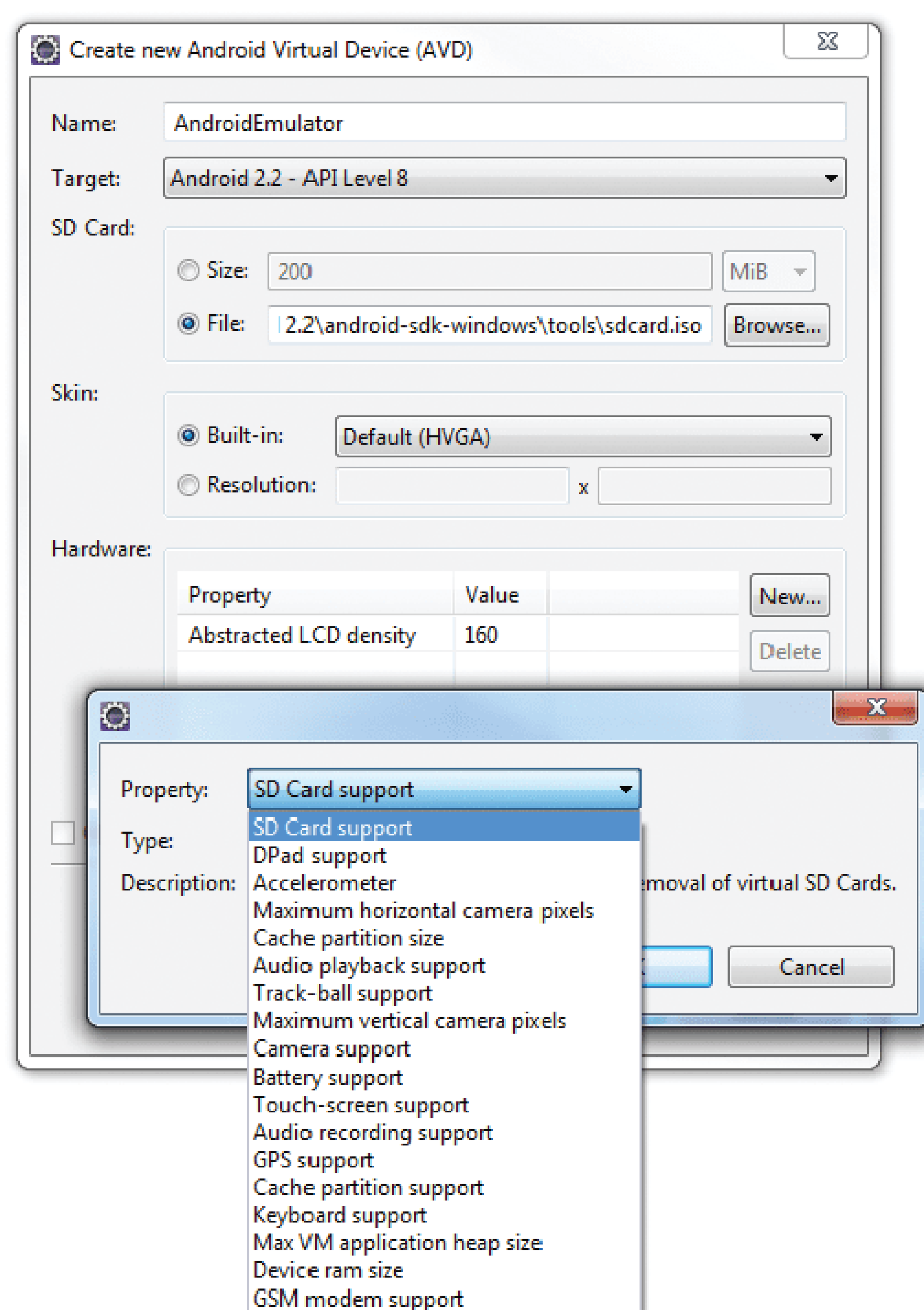


图 B-19

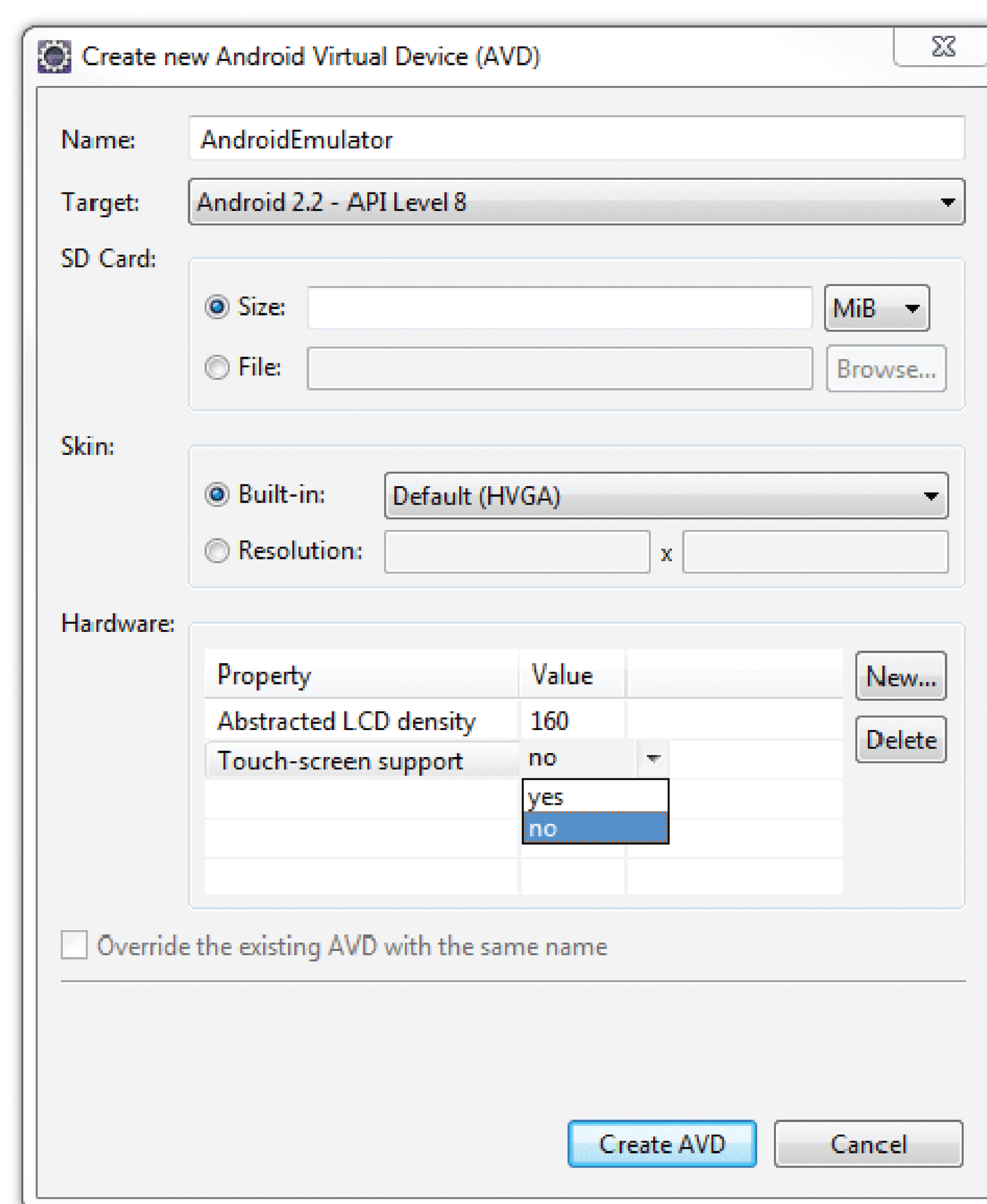


图 B-20

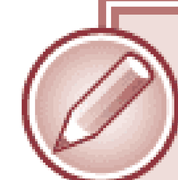
这将创建一个不带有触摸屏支持的AVD(也就是说，用户不能使用鼠标在屏幕上单击)。

还可以使用Android模拟器来模拟位置数据。第9章详细讨论过这一点。

使您的开发更有效率的一个有用的诀窍就是在开发过程中一直使Android模拟器处于运行状态——不要关闭和重启它。因为模拟器启动要花费时间，当您在调试应用程序时，最好让它一直运行着。

B.7　给模拟器发送SMS消息

使用Eclipse中可用的Dalvik Debug Monitor Service(DDMS)工具或者Telnet客户端，可以模拟发送SMS消息到Android模拟器。



注意：Telnet客户端不是Windows 7的默认安装项。要安装它，可在Windows命令提示符下输入以下命令行：

```
pkgmgr /iu:"TelnetClient"
```

键盘快捷键

Android模拟器支持多个键盘快捷键，可以使您模仿一个真正手机的行为。以下列表展示了可与模拟器一起使用的一组快捷键：

- Esc——返回
- Home——主屏幕
- F2——切换上下文敏感菜单
- F3——通话记录
- F4——锁住键盘
- F5——搜索
- F8——切换数据网络(3G)
- Ctrl+F5——提高铃声音量
- Ctrl+F6——降低铃声音量
- Ctrl+F11/Ctrl+F12——切换方向

例如，通过按下Ctrl+F11组合键，可以将模拟器改为横向模式(如图B-21所示)。



图 B-21

现在看一看在Telnet中如何做到这一点。首先，确保Android模拟器正在运行。为了Telnet到模拟器，需要知道模拟器的端口号。这可以通过查看Android模拟器窗口的标题栏获得。端口号通常以5554开始，每一个后续模拟器的端口号以2递增，例如5556、5558等。假定当前有一个Android模拟器在运行，那么可以使用以下命令Telnet到它上面：

```
C:\telnet  localhost 5554
```

要给模拟器发送一条SMS消息，可使用以下命令：

```
sms send +651234567 Hello my friend!
```

sms send命令的语法如下所示：

```
sms send <phone_number> <message>
```

图B-22展示了模拟器正在接收刚刚发送的SMS消息。

除了使用Telnet发送SMS消息之外，还可使用Eclipse中的DDMS透视图。如果DDMS透视图在Eclipse中没有显示，可以单击Open Perspective按钮(如图B-23所示)并选择Other。

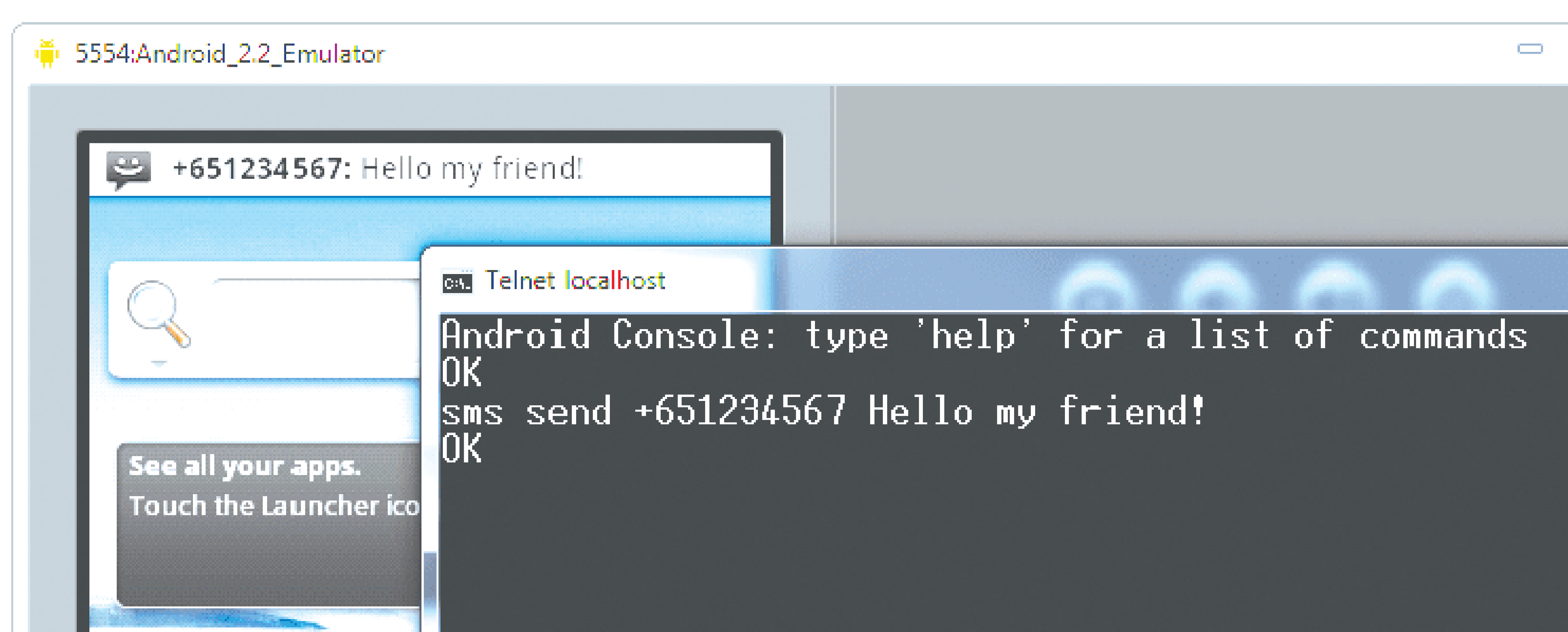


图 B-22

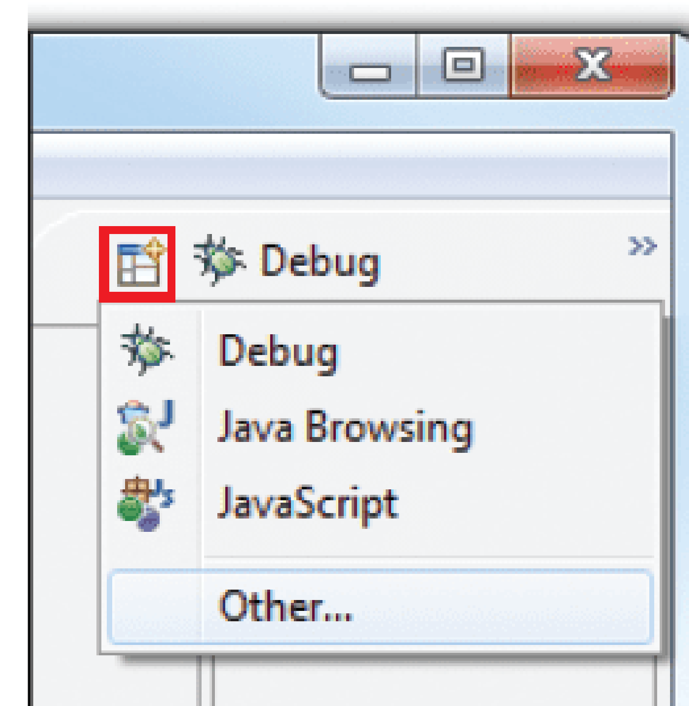


图 B-23

选择DDMS透视图(如图B-24所示)并单击OK按钮。

DDMS透视图显示之后，就可以看到Devices选项卡，里面显示了当前正在运行的模拟器的列表。选择一个打算向其发送SMS消息的模拟器实例，在Emulator Control选项卡的下面，将看到Telephony Actions部分。在Incoming number字段中，输入一个任意号码并选中SMS单选按钮。输入一条消息并单击Send按钮。

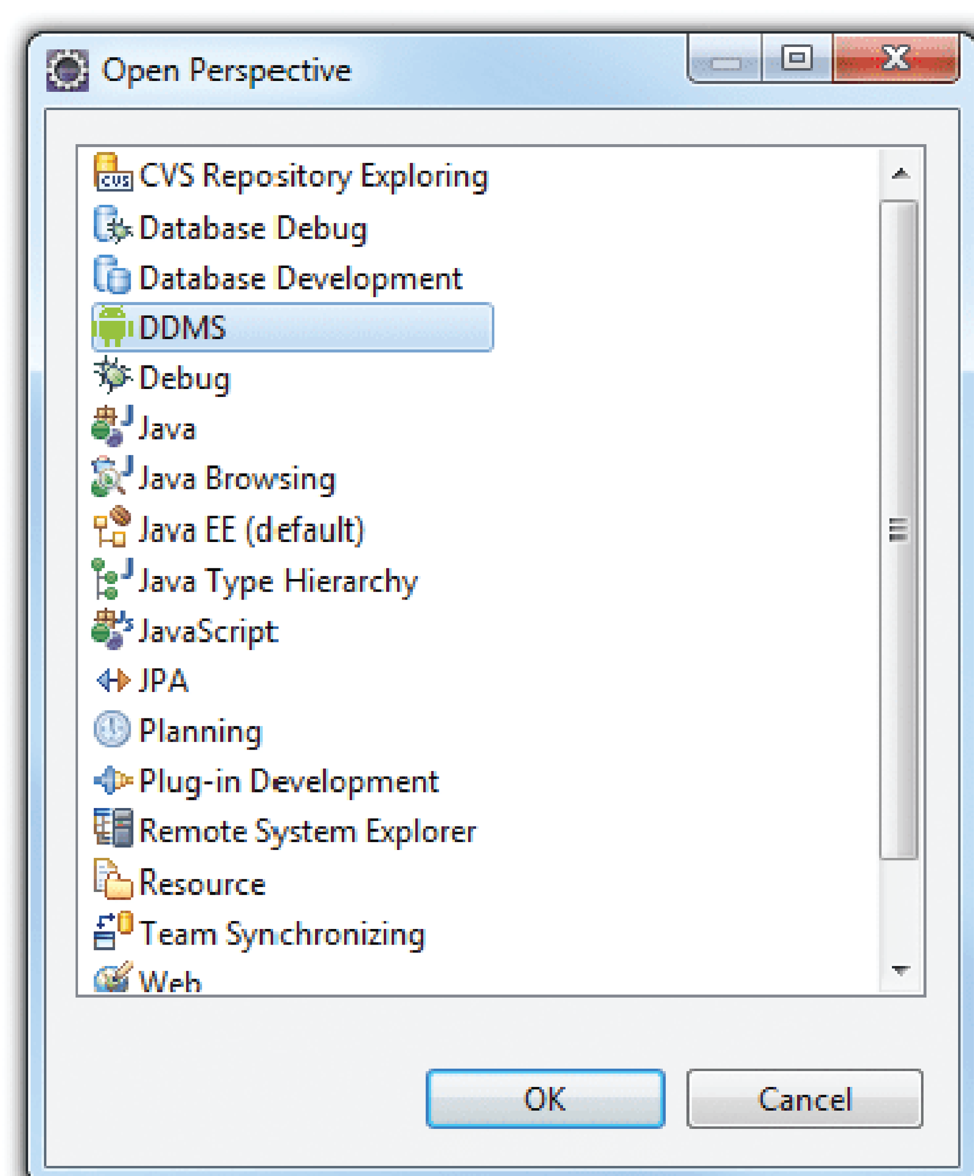


图 B-24

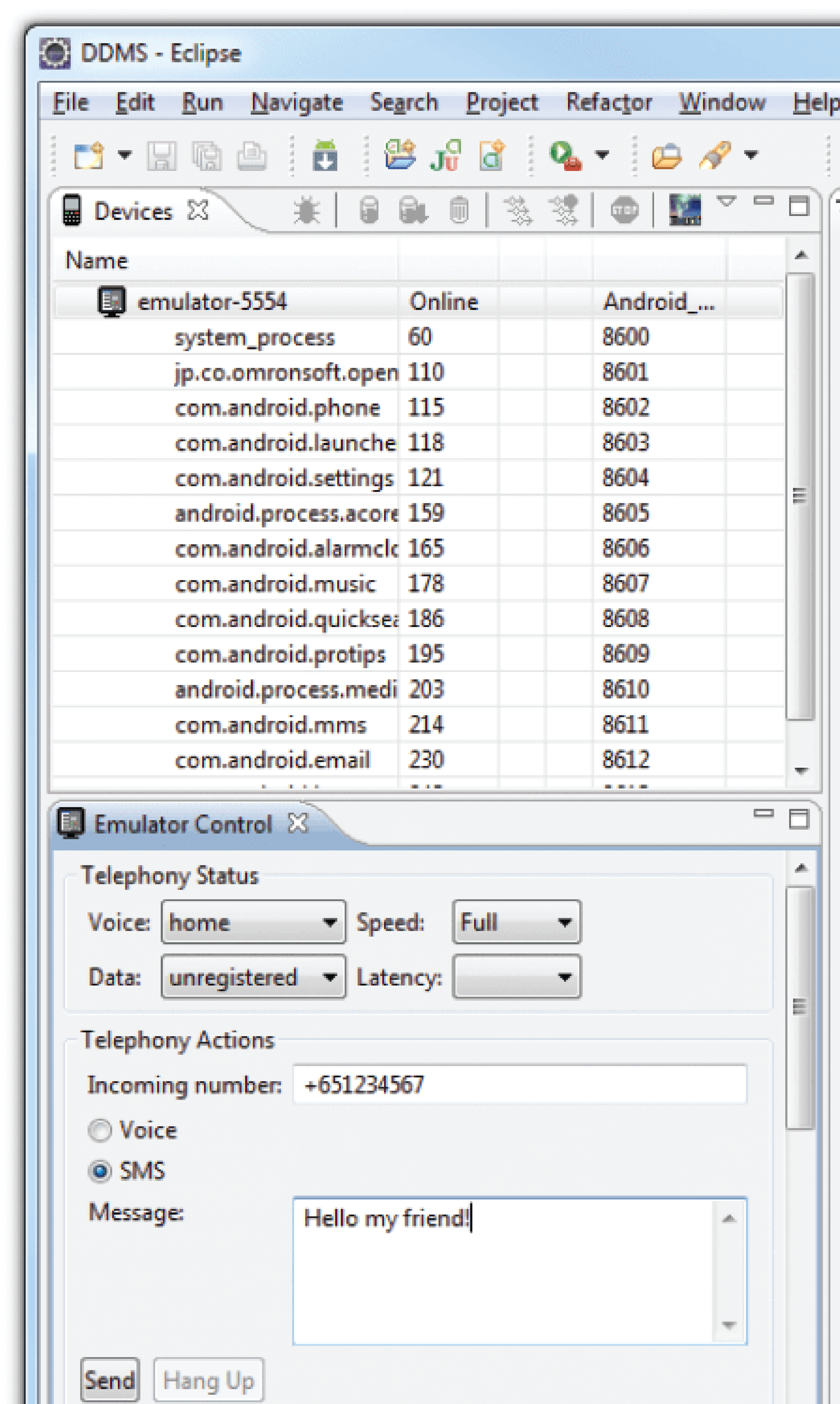


图 B-25

现在，选中的模拟器将收到一条传入的SMS消息。

如果同时有多个AVD运行，可以使用模拟器的端口号作为电话号码在每一个AVD之间发送SMS消息。例如，如果您有分别运行在端口号5554和5556上的两个模拟器，它们的电话号码将分别是5554和5556。

B.8　打电话

除了发送SMS消息到模拟器，还可以使用Telnet客户端给模拟器打电话。直接使用以下命令就可以做到这一点。

要Telnet到模拟器，使用下列命令：

```
C:\telnet localhost 5554
```

要给模拟器打电话，使用下列命令：

```
gsm call +651234567
```

gsm send命令的语法如下所示：

```
gsm call <phone_number>
```

图B-26展示了模拟器收到一个呼入的电话。

同样地，也可以使用DDMS透视图来给模拟器打电话。图B-27展示了如何使用Telephony Actions部分来打电话。

与发送SMS一样，还可以使用端口号作为电话号码在AVD之间打电话。

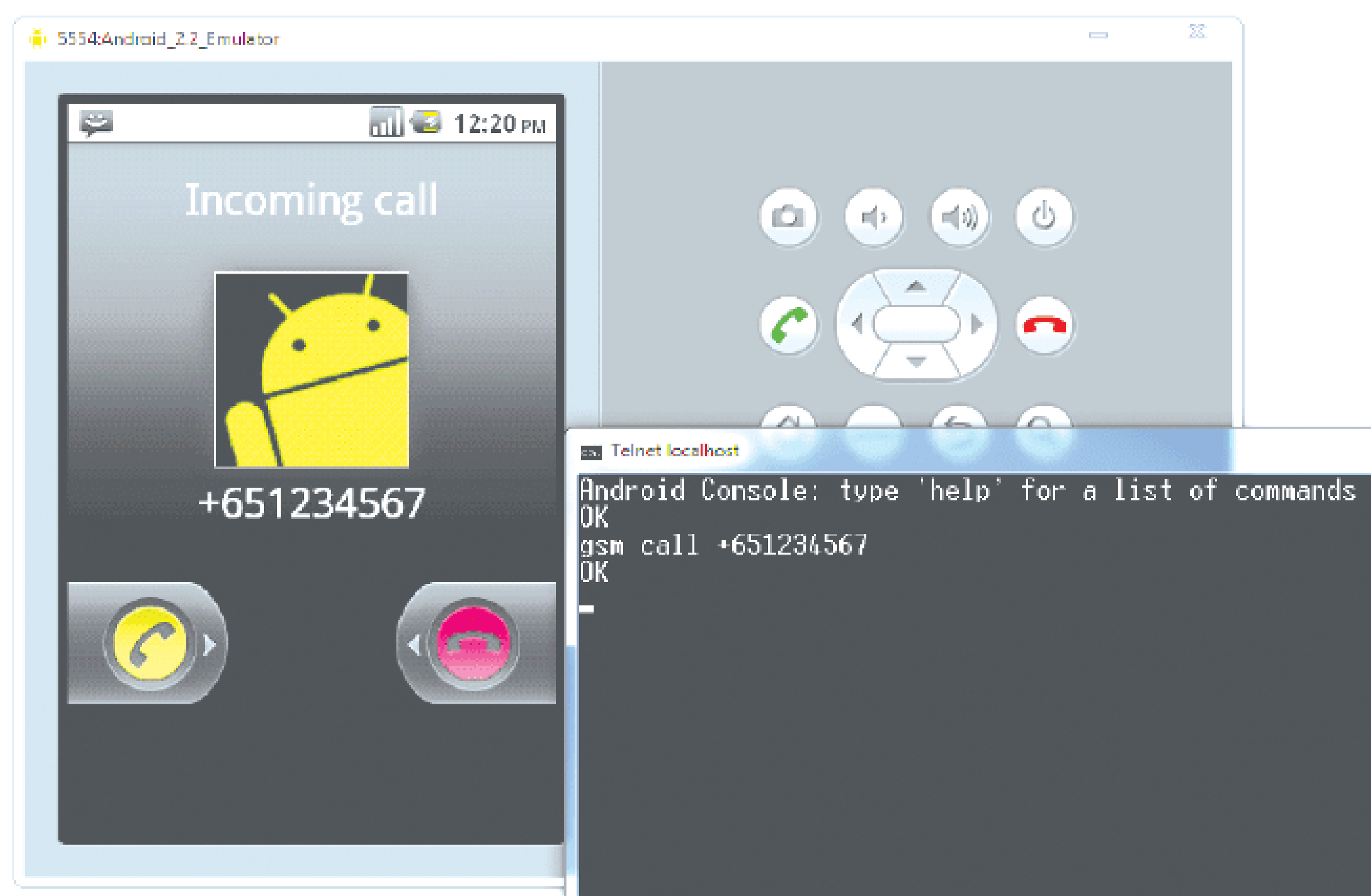


图 B-26

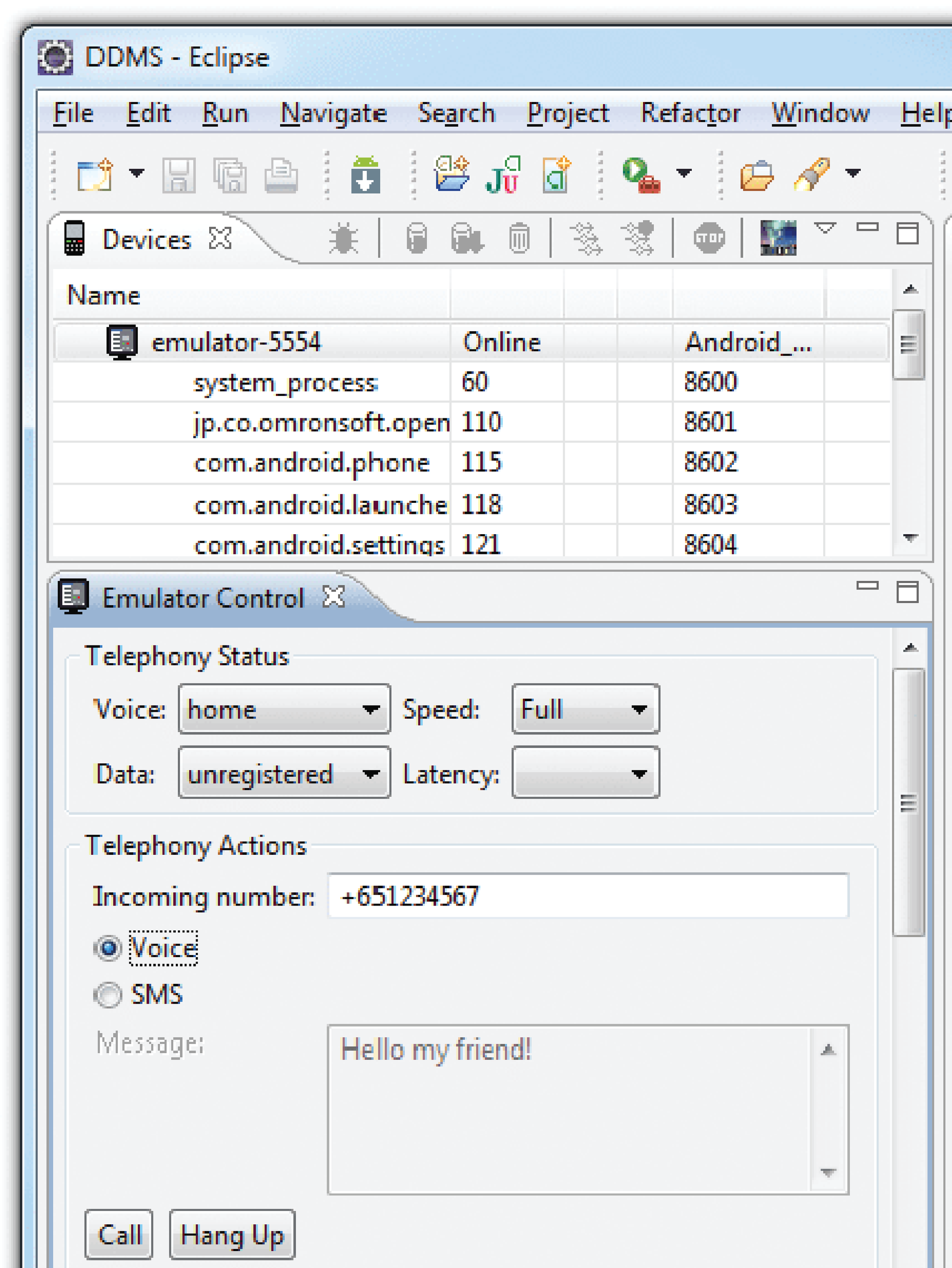


图 B-27

B.9　从模拟器中传入/出文件

偶尔，您也许需要从模拟器中传入/出文件。使用DDMS透视图是最容易的方法。从DDMS透视图中选择模拟器(或者设备，如果您有一个连接到计算机的真实的Android设备的话)并单击File Explorer选项卡来查看它的文件系统(如图B-28所示)。

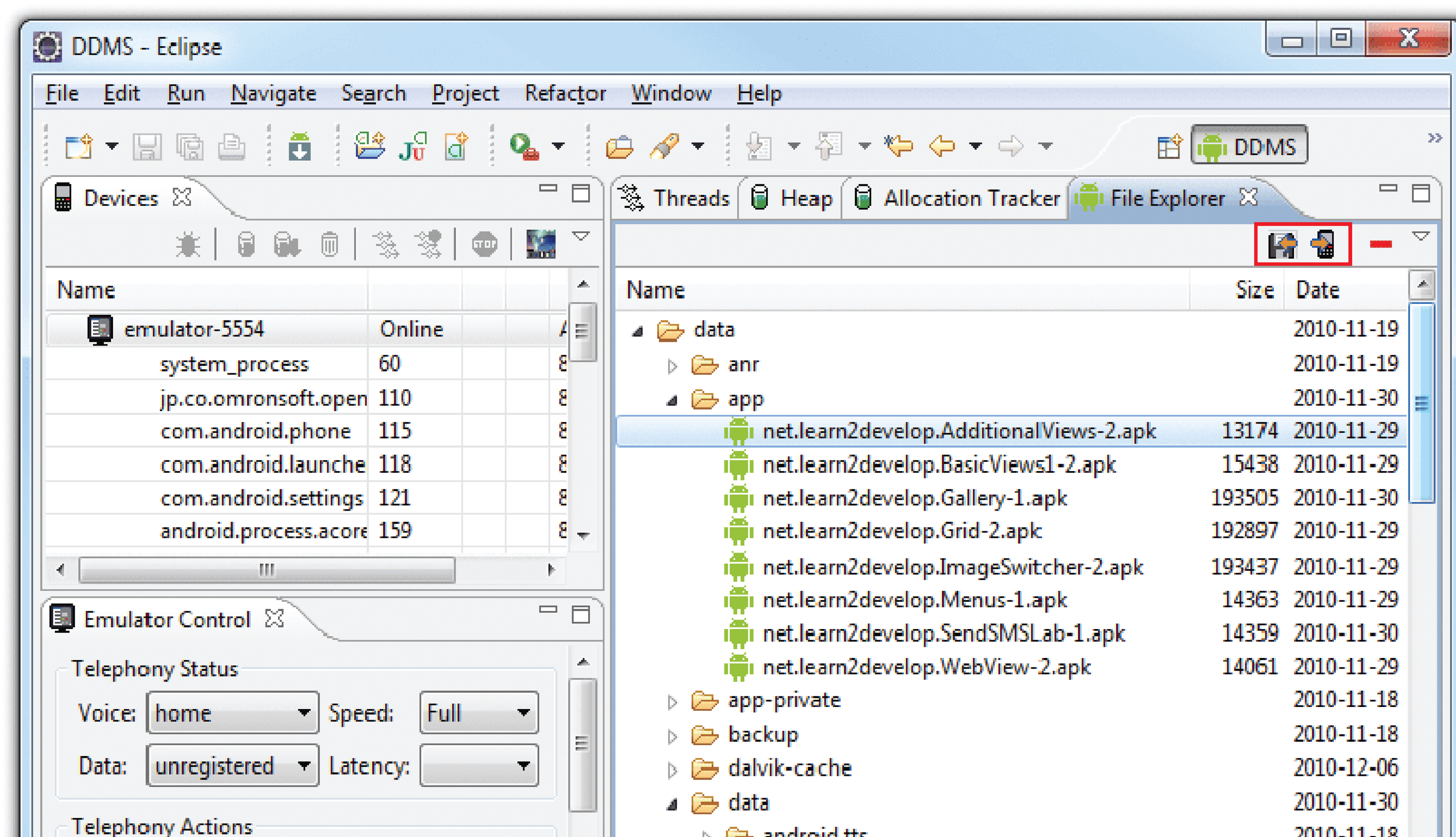


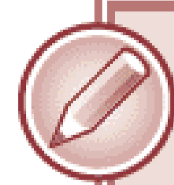
图 B-28

图B-28中显示的两个按钮可用于从模拟器中拖出或向模拟器中拖入一个文件。

或者，还可以使用Android SDK自带的adb.exe实用工具来从模拟器中拖出或拖入文件。和emulator.exe一样，这个工具位于<*Android_SDK_Folder*>\tools\文件夹下。

要从已连接的模拟器/设备上复制文件到计算机上，可使用以下命令：

```
adb.exe pull /data/app/<filename> c:\
```



注意：当使用adb.exe工具从模拟器中拖出或拖入文件时，确保只有一个AVD在运行。

图B-29展示了如何从模拟器中提取一个APK文件并将其保存到您的计算机上。

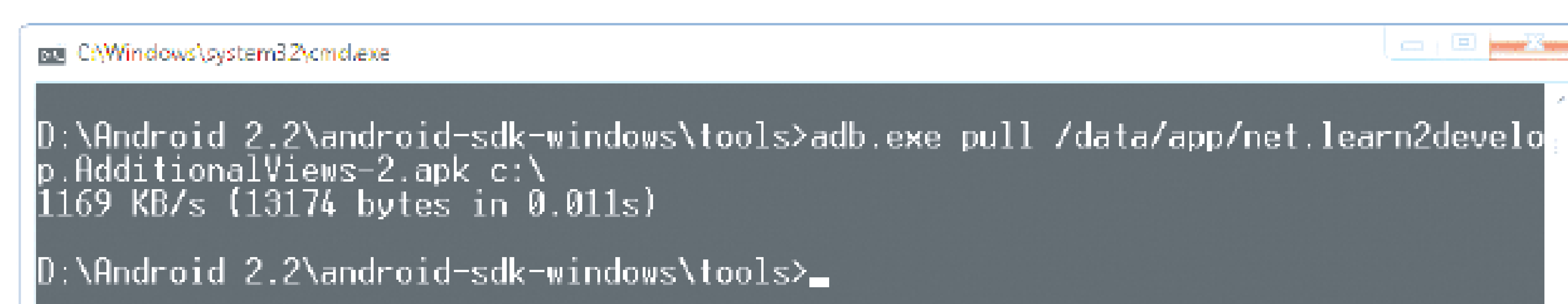


图 B-29

要将文件复制到已连接的模拟器/设备上，可使用以下命令：

```
adb.exe push NOTICE.txt /data/app
```

上面的命令复制了位于当前目录下的NOTICE.txt文件，并将其保存在模拟器的/data/app文件夹下(如图B-30所示)。

如果需要在模拟器中修改文件的权限，可以使用带有shell选项的adb.exe工具，如下所示：

```
adb.exe shell
```

图B-31展示了如何利用chmod命令修改NOTICE.txt文件的权限。

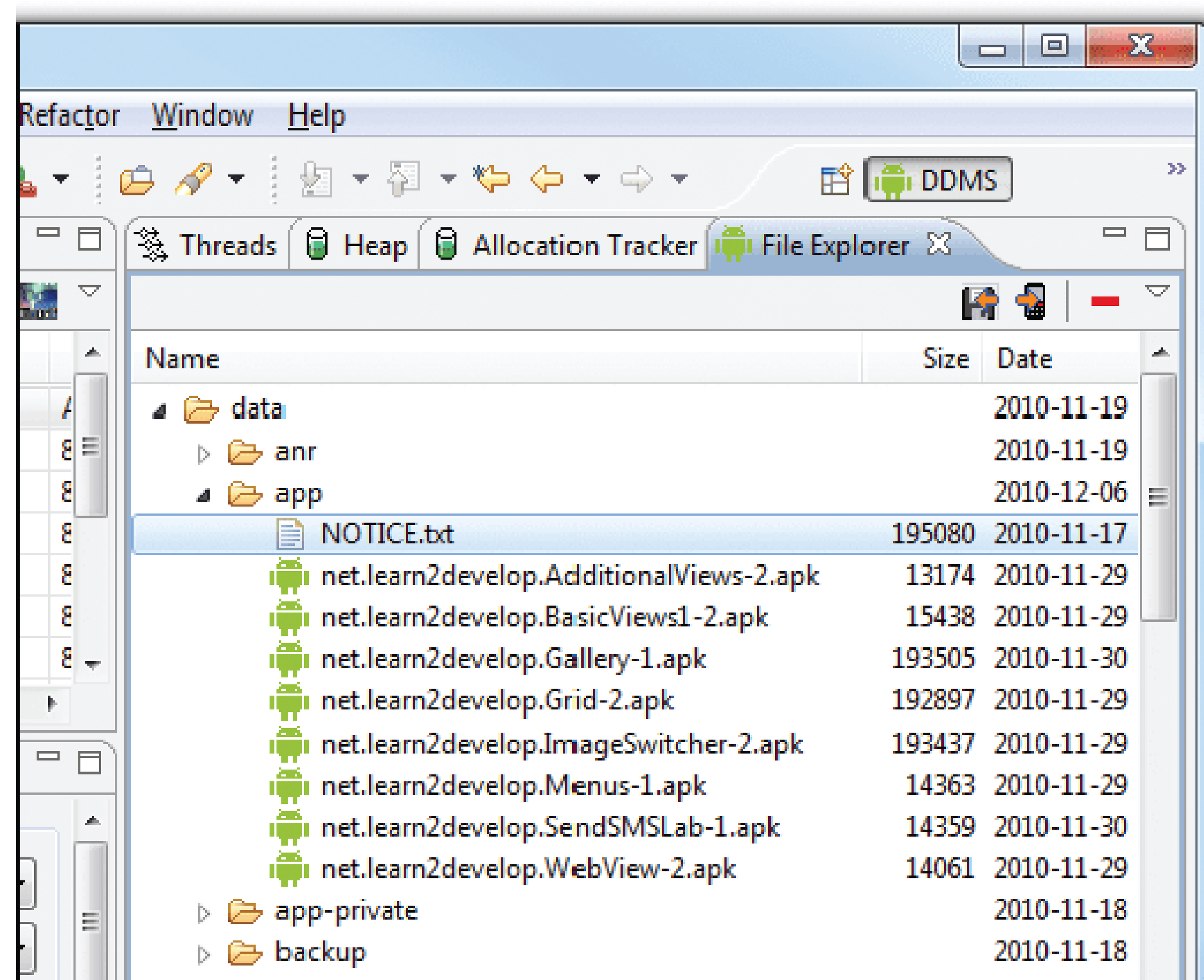


图 B-30

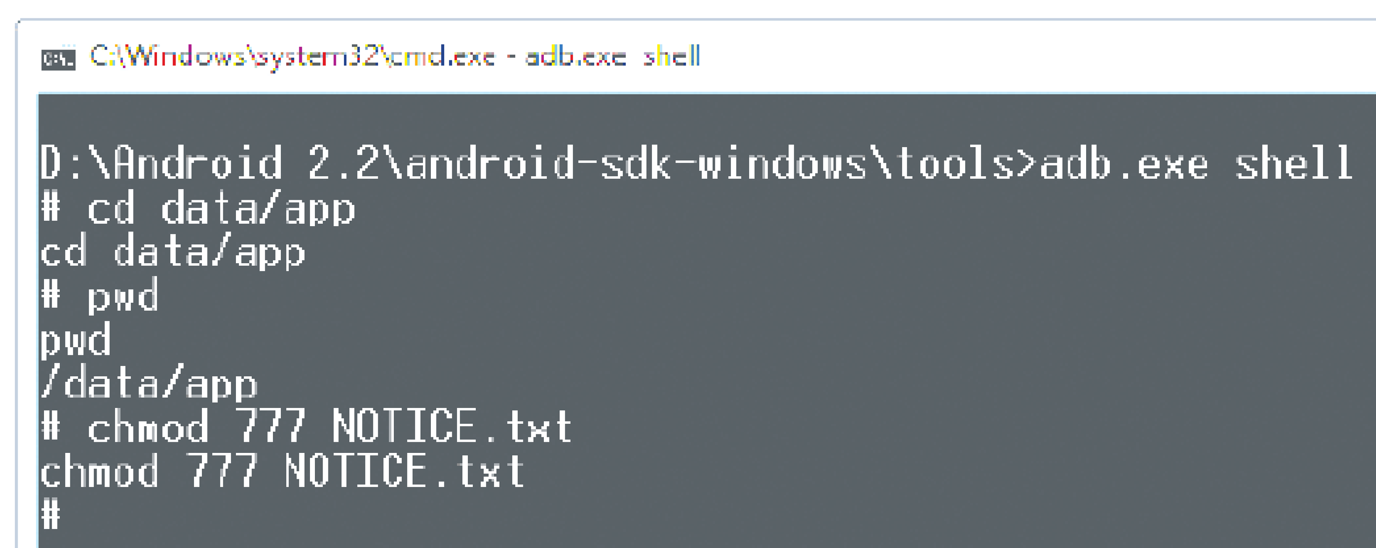


图 B-31

使用adb.exe工具，可以对Android模拟器发出Unix命令。

B.10 重置模拟器

所有部署到Android模拟器上的应用程序和文件都保存在一个名为userdata-qemu.img的文件中，此文件位于C:\Users\<*username*>\.android\avd\<*avd_name*>.avd文件夹下。例如，我有一个名为Android_2.2_Emulator的AVD，因此userdata-qemu.img文件就位于C:\Users\Wei-Meng Lee\.android\avd\Android_2.2_Emulator.avd文件夹下。

如果想将模拟器恢复到初始状态(也即重置)，只要删除userdata-qemu.img文件就行了。

附录C

练习答案

本附录包括了各章节最后的练习的答案。

C.1　第1章答案

1. AVD指的是Android虚拟设备。它代表了一个Android模拟器，可以模拟一个实际Android设备的特定配置。
2. android:versionCode属性用来以编程方式检查一个应用程序是否可以被升级。它应当包含一个顺序号(更新的应用程序的号码应该比老版本的设置得要高)。android:versionName属性主要用来显示给用户。它是一个字符串，如1.0.1。
3. string.xml文件用来存储应用程序中的所有字符串常量。这使得您可以很容易地通过替换这些字符串并重新编译应用程序来本地化您的应用程序。

C.2　第2章答案

1. Android操作系统将显示一个对话框，用户可以从中选择他们想使用的那一个活动。
2.

```
Intent i = new
    Intent(android.content.Intent.ACTION_VIEW,
      Uri.parse("http://www.amazon.com"));
startActivity(i);
```

3. 在意图筛选器中，可以指定以下内容：动作、数据、类型和类别。
4. Toast类用来向用户显示警报，并在几秒钟后消失。NotificationManager类用来在设备的状态栏上显示通知。由NotificationManager类显示的警报是持久性的，只能通过用户的选中操作来撤销。

C.3 第3章答案

1. dp单位是与密度无关的，160dp相当于1英寸。px单位对应于屏幕上的实际像素。一般应当使用dp单位，因为它可以使活动在不同屏幕大小的设备上都可以正确地缩放。
2. 随着不同屏幕大小的设备的出现，使用AbsoluteLayout使得您的应用程序在跨设备应用时很难保持一致的外观和体验。
3. 当一个活动被终止或转入后台时，将触发onPause()事件。onSaveInstanceState()事件与onPause()事件类似，除了它不总是被调用，例如当用户按下Back按钮来终止活动时。
4. 3个事件是onPause()、onSaveInstanceState()和onRetainNonConfigurationInstance()。

C.4 第4章答案

1. 应该检验每一个RadioButton的isChecked()方法来确定其是否被选中。
2. 可以使用getResources()方法。
3. 以下代码片段用于获取当前日期：

```
//---获取当前日期---
Calendar today = Calendar.getInstance();
yr = today.get(Calendar.YEAR);
month = today.get(Calendar.MONTH);
day = today.get(Calendar.DAY_OF_MONTH);
showDialog(DATE_DIALOG_ID);
```

C.5 第5章答案

1. ImageSwitcher可以使图像动画显示。您可以在图像显示时以及被另一幅图像替换时动画显示它。
2. 两个方法是onCreateOptionsMenu()和onOptionsItemSelected()。
3. 两个方法是onCreateContextMenu()和onContextItemSelected()。
4. 要防止启动设备的Web浏览器，需要实现WebViewClient类并重写shouldOverrideUrl-Loading()方法。

C.6 第6章答案

1. 前者允许在一个应用程序中的所有活动之间共享数据，而后者只允许创建它的活动进行访问。
2. 方法名称是getExternalStorageDirectory()。
3. 权限是WRITE_EXTERNAL_STORAGE。

C.7 第7章答案

1. 代码如下所示：

```
Cursor c = managedQuery(
    allContacts,
    projection,
    ContactsContract.Contacts.DISPLAY_NAME + " LIKE ?",
    new String[] {"%jack%"} ,
    ContactsContract.Contacts.DISPLAY_NAME + " ASC");
```

2. 方法是getType()、onCreate()、query()、insert()、delete()和update()。

3. 代码如下所示：

```
<provider android:name="BooksProvider"
        android:authorities="net.learn2develop.provider.Books" />
```

C.8 第8章答案

1. 可以以编程方式从Android应用程序中发送SMS消息，也可以以应用程序的名义调用内置的Messaging应用程序来发送SMS信息。

2. 两个权限是SEND_SMS和RECEIVE_SMS。

3. 广播接收者应该触发一个将由活动接收的新意图。活动应该实现另一个BroadcastReceiver来侦听这个新的意图。

4. 权限是INTERNET。

C.9 第9章答案

1. 可能的原因如下所示：
- 没有Internet连接
- <uses-library>元素在AndroidManifest.xml文件中的位置错误
- AndroidManifest.xml文件中缺少INTERNET权限

2. 地理编码是将一个地址转换成其坐标(经度和纬度)。反向地理编码是将一对位置坐标转换成一个地址。

3. 两个位置服务提供商如下所示：
- LocationManager.GPS_PROVIDER
- LocationManager.NETWORK_PROVIDER

4. 方法是addProximityAlert()。

C.10 第10章答案

1. 这是因为服务和主调活动运行在同一个进程上。如果服务是长时间运行的，那么需要在一个单独的线程上运行它，这样才不会阻塞活动。

2. IntentService类与Service类相似，除了前者在一个单独的线程上运行任务，并且当任务结束执行时可以自动停止服务。
3. 3个方法是doInBackground()、onProgressUpdate()和onPostExecute()。
4. 服务可以广播一个意图，而活动可以使用一个IntentFilter类来注册一个意图。

C.11　第11章答案

1. 在AndroidManifest.xml文件中使用minSdkVersion属性来指定所需的Android最低版本。
2. 可以使用Java SDK的keytool.exe实用工具，也可以使用Eclipse的Export功能来生成证书。
3. 转到Settings应用程序并选择Application项。选中Unknown sources项。